



SunSHIELD 基本セキュリティ モジュール

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

Part No: 805-5853-10
1998 年 11 月

本製品およびそれに関連する文書は著作権法により保護されており、その使用、複製、頒布および逆コンパイルを制限するライセンスのもとにおいて頒布されます。日本サン・マイクロシステムズ株式会社による事前の許可なく、本製品および関連する文書のいかなる部分も、いかなる方法によっても複製することが禁じられます。

本製品の一部は、カリフォルニア大学からライセンスされている Berkeley BSD システムに基づいていることがあります。UNIX は、X/Open Company, Ltd. が独占的にライセンスしている米国ならびに他の国における登録商標です。フォント技術を含む第三者のソフトウェアは、著作権により保護されており、提供者からライセンスを受けているものです。

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

本製品に含まれる HG 明朝 L と HG ゴシック B は、株式会社リコーがリョービイマジクス株式会社からライセンス供与されたタイプフェイスマスタをもとに作成されたものです。平成明朝体 W3 は、株式会社リコーが財団法人 日本規格協会 文字フォント開発・普及センターからライセンス供与されたタイプフェイスマスタをもとに作成されたものです。また、HG 明朝 L と HG ゴシック B の補助漢字部分は、平成明朝体 W3 の補助漢字を使用しています。なお、フォントとして無断複製することは禁止されています。

Sun, Sun Microsystems, SunSoft, SunDocs, SunExpress, OpenWindows, SunSHIELD, SHIELD は、米国およびその他の国における米国 Sun Microsystems, Inc. (以下、米国 Sun Microsystems 社とします) の商標もしくは登録商標です。

サンのロゴマークおよび Solaris は、米国 Sun Microsystems 社の登録商標です。

すべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

OPENLOOK, OpenBoot, JLE は、日本サン・マイクロシステムズ株式会社の登録商標です。

Wnn は、京都大学、株式会社アステック、オムロン株式会社で共同開発されたソフトウェアです。

Wnn6 は、オムロン株式会社で開発されたソフトウェアです。(Copyright OMRON Co., Ltd. 1998 All Rights Reserved.)

ATOK は、株式会社ジャストシステムの登録商標です。

ATOK7 は株式会社ジャストシステムの著作物であり、ATOK7 にかかる著作権その他の権利は、すべて株式会社ジャストシステムに帰属します。

ATOK8 は株式会社ジャストシステムの著作物であり、ATOK8 にかかる著作権その他の権利は、すべて株式会社ジャストシステムに帰属します。

本書で参照されている製品やサービスに関しては、該当する会社または組織に直接お問い合わせください。

OPEN LOOK および Sun Graphical User Interface は、米国 Sun Microsystems 社が自社のユーザおよびライセンス実施権者向けに開発しました。米国 Sun Microsystems 社は、コンピュータ産業用のビジュアルまたはグラフィカル・ユーザインタフェースの概念の研究開発における米国 Xerox 社の先駆者としての成果を認めるものです。米国 Sun Microsystems 社は米国 Xerox 社から Xerox Graphical User Interface の非独占的ライセンスを取得しており、このライセンスは米国 Sun Microsystems 社のライセンス実施権者にも適用されます。

DiComboBox ウィジェットと DtSpinBox ウィジェットのプログラムおよびドキュメントは、Interleaf, Inc. から提供されたものです。(Copyright (c) 1993 Interleaf, Inc.)

本書は、「現状のまま」をベースとして提供され、商品性、特定目的への適合性または第三者の権利の非侵害の黙示の保証を含みそれに限定されない、明示的であるか黙示的であるかを問わない、なんらの保証も行われぬものとします。

本製品が、外国為替および外国貿易管理法 (外為法) に定められる戦略物資等 (貨物または役務) に該当する場合、本製品を輸出または日本国外へ持ち出す際には、日本サン・マイクロシステムズ株式会社の事前の書面による承諾を得ることのほか、外為法および関連法規に基づく輸出手続き、また場合によっては、米国商務省または米国所轄官庁の許可を得ることが必要です。

原典: SunSHIELD Basic Security Module Guide

Part No: 805-2635-10

Revision A, September 1998

© 1998 by Sun Microsystems, Inc.



目次

はじめに ix

1. インストール 1

BSM を使用可能にする方法 2

BSM を使用不可にする方法 2

BSM とクライアントサーバの関係 3

2. 監査機能の管理 5

監査機能の概要 6

監査機能の起動 6

監査クラスとイベント 7

カーネルイベント 7

ユーザレベルのイベント 7

監査レコード 8

監査フラグ 8

監査フラグの定義 9

監査フラグの構文 10

設定済みの監査フラグを変更する接頭辞 11

audit_controlファイル 12

audit_control ファイルのサンプル 13

audit_user ファイル内のユーザ監査フィールド 13

| | |
|---------------------------------------|----|
| プロセスの監査特性 | 15 |
| プロセス事前選択マスク | 15 |
| 監査 ID | 15 |
| 監査セッション ID | 16 |
| 端末 ID | 16 |
| 監査トレールの作成 | 16 |
| audit_data ファイル | 17 |
| 監査デーモンの役割 | 17 |
| 監査デーモンが使用できるディレクトリ | 18 |
| 監査ファイルを常に管理可能な状態に保つ | 18 |
| audit_warn スクリプト | 19 |
| auditreduce コマンドの使用方法 | 20 |
| 監査コストの制御 | 23 |
| 処理時間の増大に伴うコスト | 23 |
| 分析に伴うコスト | 23 |
| 格納に伴うコスト | 24 |
| 通常のユーザの監査 | 25 |
| 効率的な監査 | 25 |
| ▼ audit ファイルを組み合わせて縮小する方法 | 26 |
| 監査トレールについて | 27 |
| 監査ファイルの詳細 | 28 |
| not_terminated マークが付いたアクティブでないファイルの処理 | 30 |
| ▼ 監査パーティションを作成してエクスポートする方法 | 31 |
| ▼ 監査を構成する方法 | 33 |
| ▼ 監査構成の計画を立てる方法 | 34 |
| 監査トレールのオーバーフローを防ぐ | 37 |
| ▼ 監査トレールのオーバーフローを防ぐ方法 | 38 |

| | |
|------------------------------|----|
| auditconfig コマンド | 38 |
| 監査方針の設定 | 40 |
| ▼ どのイベントがどの監査クラスに属するかを変更する方法 | 41 |
| クラス定義の変更 | 42 |
| 3. 監査トレールの分析 | 43 |
| 監査機能 | 43 |
| 監査ユーザ ID | 44 |
| 監査セッション ID | 44 |
| それ自身で意味のわかる監査レコード | 44 |
| 監査レコードのマージ、選択、表示、変換に使用するツール | 44 |
| 監査レコードの形式 | 45 |
| 監査トークンの順序 | 46 |
| ユーザが読める監査レコード書式 | 46 |
| header トークン | 47 |
| trailer トークン | 48 |
| arbitrary トークン | 48 |
| arg トークン | 49 |
| attr トークン | 49 |
| exit トークン | 50 |
| file トークン | 50 |
| groups トークン | 50 |
| in_addr トークン | 50 |
| ip トークン | 51 |
| ipc トークン | 51 |
| ipc_perm トークン | 51 |
| iport トークン | 52 |
| opaque トークン | 52 |
| path トークン | 52 |

| | | |
|-----------|--------------------------------|-----------|
| | process トークン | 53 |
| | return トークン | 53 |
| | seq トークン | 53 |
| | socket トークン | 54 |
| | subject トークン | 54 |
| | text トークン | 55 |
| | auditreduce コマンドの使用法 | 55 |
| | 分散システムで auditreduce を役立てる方法 | 56 |
| | auditreduce の使用法 | 56 |
| | その他の有用な auditreduce オプション | 57 |
| | praudit の使用法 | 58 |
| 4. | デバイスの割り当て | 61 |
| | デバイスの使用に伴うリスク | 62 |
| | デバイス割り当て機構の構成要素 | 62 |
| | デバイス割り当てユーティリティの使用法 | 63 |
| | 割り当てエラー状態 | 64 |
| | device_maps ファイル | 65 |
| | device_allocate ファイル | 66 |
| | デバイスクリーンスクリプト | 69 |
| | オブジェクトの再使用 | 69 |
| | 新しいデバイスクリーンスクリプトの作成 | 71 |
| | ロックファイルの設定 | 71 |
| | ▼ 割り当て可能にするデバイスのロックファイルを設定する方法 | 71 |
| | デバイスの管理と追加 | 74 |
| | ▼ デバイスを管理する方法 | 74 |
| | ▼ 新しい割り当て可能デバイスを追加する方法 | 74 |
| | デバイス割り当ての使用法 | 75 |
| | ▼ デバイスを割り当てる方法 | 76 |

| | |
|----------------------|----|
| ▼ デバイスの割り当てを解除する方法 | 76 |
| A. 監査レコードの説明 | 77 |
| 監査レコードの構造 | 77 |
| 監査トークンの構造 | 78 |
| acl トークン | 80 |
| arbitrary トークン | 80 |
| arg トークン | 81 |
| attr トークン | 82 |
| exec_args トークン | 82 |
| exec_env トークン | 83 |
| exit トークン | 83 |
| file トークン | 84 |
| groups トークン (使用しません) | 84 |
| header トークン | 85 |
| in_addr トークン | 86 |
| ip トークン | 86 |
| ipc トークン | 87 |
| ipc_perm トークン | 88 |
| iport トークン | 88 |
| newgroups トークン | 89 |
| opaque トークン | 90 |
| path トークン | 90 |
| process トークン | 90 |
| return トークン | 91 |
| seq トークン | 92 |
| socket トークン | 92 |
| socket-inet トークン | 93 |
| subject トークン | 93 |

| | |
|----------------------|------------|
| text トークン | 94 |
| trailer トークン | 94 |
| 監査レコード | 95 |
| 一般的な監査レコードの構造 | 96 |
| カーネルレベルで生成される監査レコード | 96 |
| ユーザレベルで生成される監査レコード | 181 |
| イベントからシステムコールへの変換 | 197 |
| B. BSM リファレンス | 213 |
| 索引 | 217 |

はじめに

Solaris™ の SHIELD™ 基本セキュリティモジュール (Basic Security Module、BSM) は、Trusted Computer Evaluation Criteria (TCSEC) で C2 レベルとして定義されたセキュリティ機能 (標準 UNIX では提供されません) を提供します。BSM が提供する機能は、セキュリティ監査サブシステム、および、取り外し可能な、または割り当て可能なデバイスでオブジェクトを再使用できるデバイス割り当て機構です。C2 離散アクセス制御は、C2 識別および認証機能とともに標準 Solaris システムによって提供されます。

対象読者

このマニュアルは、BSM の設定と管理を担当するシステム管理者を対象にしています。システム管理の基本概念とテキストエディタの使用方法に精通しておくに役に立ちます。

このマニュアルの構成

第 1 章では、BSM を使用可能にする方法と使用不可能にする方法を説明します。この章では Solaris システムを使用可能にして、セキュリティ機能を使用する方法と、BSM を使用する環境においてクライアントとサーバが相互に動作する方法とを取り上げます。

第 2 章では、システム管理と監査サブシステムの構成を説明します。この章では監査トレール記憶デバイスの管理、グローバルでユーザ単位の事前選択、サイト固有の構成オプションの設定を取り上げます。

第 3 章では、監査トレール分析の処理と事後処理について詳細に説明します。この章では監査レコードの構造と書式全般、監査トレール印刷ユーティリティ、監査レコード選択およびマージユーティリティを取り上げます。

第 4 章では、取り外し可能、または割り当て可能なデバイスのための割り当て機構を説明します。この章では、割り当て可能なデバイスファイルの設定と管理、特権を持たないユーザによる割り当て機構の使用方法を取り上げます。

付録 A では、生成される監査レコードの内容について詳細に説明します。

付録 B では、Solaris SHIELD 基本セキュリティモジュール用に追加されたマニュアルページを一覧表にして説明します。

マニュアルの注文方法

SunDocs™ プログラムでは、米国 Sun Microsystems™, Inc. (以降、Sun™ とします) 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、ディレクトリ名、画面上のコンピュータ出力、または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。 system% |
| 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は区別して示します。 | system% su password: |
| <i>AaBbCc123</i> | 変数を示します。実際に使用する特定の名称または値で置き換えます。 | ファイルを削除するには、rm <i>filename</i> と入力します。 |
| 『 』 | 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、節、ボタンやメニュー名、または強調する単語を示します。 | 第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。 |
| \ | 枠で囲まれたコード例で、テキストがページ行幅を越える場合、バックスラッシュは継続を示します。 | sun% grep '^#define \ XV_VERSION_STRING' |

ただし AnswerBook2™ では、ユーザー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。

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 y|n [filename]
```

■ Bourne シェルおよび Korn シェルのプロンプト

```
system$ command y|n [filename]
```

■ スーパーユーザーのプロンプト

```
system# command y|n [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。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- 「x86」という用語は、一般に Intel 8086 ファミリに属するマイクロプロセッサを意味します。これには、Pentium、Pentium Pro の各プロセッサ、および AMD と Cyrix が提供する互換マイクロプロセッサチップが含まれます。このマニュアルでは、このプラットフォームのアーキテクチャ全体を指すときに「x86」という用語を使用し、製品名では「Intel 版」という表記で統一しています。

インストール

BSM は Solaris 2.3 以降の全リリースに含まれ、リリース媒体により提供されています。BSM は、簡単な 2 つのスクリプトのうちの一方を実行することにより使用可能、あるいは使用不可にできるようになっているので、別個にインストールする必要はありません。次のパッケージをインストールすれば、BSM ソフトウェアはすべて初期システムインストールに含まれます。

- SUNWcar – コアアーキテクチャー
- SUNWcsr – コア SPARC
- SUNWcsu – コア SPARC
- SUNWhea – ヘッドファイル
- SUNWman – オンラインマニュアル ページ

次の作業は `root` でのみ実行するようにします。また、コマンドはサーバまたはスタンドアロンシステム上でのみ実行するようにし、ディスクレスクライアント上では絶対に実行しないでください。

- 2ページの「BSM を使用可能にする方法」
- 2ページの「BSM を使用不可にする方法」
- 3ページの「BSM とクライアントサーバの関係」

BSM を使用可能にする方法

root になったら、telinit を使用してシステムをシングルユーザモードにします (init(1M) のマニュアルページを参照)。

```
# /etc/telinit 1
```

シングルユーザモードでディレクトリを /etc/security/audit ディレクトリに変更し、そこにある bsmconv スクリプトを実行します。このスクリプトによって、リポート後に標準 Solaris マシンが設定され、BSM が実行されます。

```
# cd /etc/security
# ./bsmconv
```

スクリプトを作成したら、telinit コマンドを使ってシステムを停止させます。次に、システムをリポートし、マルチユーザ BSM システムとして起動します。

```
# /etc/telinit 6
```

注 - bsmconv スクリプトは、Stop-A キーボードシーケンスでシステムをアポートさせる機能を停止するための行を、/etc/system に追加します。Stop-A キーボードシーケンスでシステムをアポートさせる機能を使用し続けたいときは、/etc/system 内の "set abort_enable = 0" という行をコメント化してください。

BSM を使用不可にする方法

BSM がある時点で不要になった場合は、bsmunconv スクリプトを実行して使用不可にできます (bsmconv(1M) のマニュアルページを参照)。もう一度 telinit コマンドを使用してシステムをシングルユーザモードしてからディレクトリを /etc/security ディレクトリに変更し、bsmunconv スクリプトを実行します。

```
# /etc/telinit 1
# cd /etc/security
# ./bsmunconv
```

「BSM を使用不可にしたあと」、システムをリポートし、マルチユーザ Solaris マシンとして起動します。

```
# /etc/telinit 6
```

注 - bsunmconvスクリプトは、Stop-A キーボードシーケンスでシステムをアボートさせる機能を停止するための行を、`/etc/system` から削除します。bsunconvスクリプトの実行後も Stop-A キーボードシーケンスでシステムをアボートさせる機能を停止させたいときは、`/etc/system` 内に `"set abort_enable = 0"` という行を再入力してください。

BSM とクライアントサーバの関係

Solaris 2.1 リリースでは、BSM を使用できるシステムにディスククライアントを追加したり、そこから削除したりするために、別の2つの手順が必要でした。Solaris 2.3 リリースからは BSM が組み込まれているため、この2つの手順は必要ありません。BMS をサーバ上で使用可能にすると、そのサーバのすべてのクライアント上で BSM 機能が自動的に使用可能になります。

監査機能の管理

この章では監査機能を設定し、管理する方法を説明します。監査機能により、システム管理者はユーザの動作を監視できます。管理者は監査機構を使用してセキュリティが破られる可能性を検出できます。監査機能は、システムの使用について不審なパターンや異常なパターンを発見し、特定のユーザに関する疑わしい動作を追跡する方法を提供します。また、監査は抑止力としても機能します。つまり、ユーザは自分の動作が監査されそうだと考えると、違反行為を思いとどまる可能性が大きくなります。

- 6ページの「監査機能の概要」
- 6ページの「監査機能の起動」
- 7ページの「監査クラスとイベント」
- 8ページの「監査フラグ」
- 15ページの「プロセスの監査特性」
- 16ページの「監査トレールの作成」
- 23ページの「監査コストの制御」
- 25ページの「通常のユーザの監査」
- 25ページの「効率的な監査」
- 27ページの「監査トレールについて」
- 37ページの「監査トレールのオーバーフローを防ぐ」
- 40ページの「監査方針の設定」
- 42ページの「クラス定義の変更」

監査機能の概要

監査を正常に機能させるには、識別と認証という2つのセキュリティ機能が重要な役割を果たします。ログイン時にユーザがユーザ名とパスワードを与えると、固有の監査 ID がそのユーザのプロセスに関連付けられます。監査 ID は、ログインセッション中に起動されるすべてのプロセスによって継承されます。ユーザが ID を変更しても (su (1M) のマニュアルページを参照)、実行するすべての動作は同じ監査 ID によって追跡されます。

監査機能によって次のことが可能になります。

- システム上で発生するセキュリティに関するイベントを監視する
- イベントを監査トレールに記録する
- 誤用または権限のない動作を (監査トレールの分析により) 検出する

システム管理者は、システムの構成時にどの動作を監視するかを選択します。管理者は各ユーザに行う監査の程度を細かく調整することもできます。

監査データを収集したら、監査ファイル縮小ツールと変換ツールによって監査トレールの注目すべき部分を検査できます。たとえば、個別のユーザまたはグループの監査レコードを調べるか、特定の日に発生した特定のタイプのイベントのレコードをすべて調べるか、または、特定の日に発生したレコードだけを取り出すかを選択できます。

この章では、監査機能を設定し、管理する方法を説明します。監査データを変換する方法については、第4章を参照してください。

監査機能の起動

`/usr/sbin/auditd` を `root` として実行して監査デーモンを起動すると、監査機能が使用可能になります (`auditd(1M)` のマニュアルページを参照)。

パス名 `/etc/security/audit_startup` を持つファイルがあると、システムがマルチユーザモードになったときに監査デーモンが自動的に実行されます。実際には、このファイルは監査デーモンを実行する前に起動シーケンスの一部として起動される実行可能スクリプトです (`audit_startup(1M)` のマニュアルページを参照)。イベントを自動的にクラス別に構成して、監査方針を設定するデフォルトの `audit_startup` スクリプトは BSM パッケージのインストール時に設定されます。

監査クラスとイベント

セキュリティに関わりを持つ動作について監査を行います。監査可能なシステム動作は監査イベントとして `etc/security/audit_event` ファイル内に定義します。各監査可能イベントは、記号名、イベント番号、事前選択クラスのセット、および短い説明によって定義します (`audit_event(4)` のマニュアルページを参照)。

ほとんどのイベントは個々のユーザの動作が原因で発生します。しかし、一部のイベントはカーネル割り込みレベルで発生したり、ユーザが識別され認証される前に発生したりするので、ユーザに帰因しないものもあります。ユーザに帰因しないイベントも監査されます。

各監査イベントは、1 つまたは複数の監査クラスに属するものとして定義します。イベントをクラスに割り当てると、管理者はより多くのイベントを、より簡単に処理できるようになります。イベントをクラスに割り当てると、管理者はより多くのイベントを、より簡単に処理できるようになります。また、クラス自体も変更できます。これらの変更は `audit_event` ファイル内で行います。

管理者が特定のイベントを含むクラスを監査のために事前選択することによって、監査可能なイベントが監査トレールに記録されます。32 の監査クラスのうち、18 のクラスを定義します。この 18 のクラスには 2 つのグローバルクラス、`all` と `no` が含まれます。

カーネルイベント

カーネルによって生成されるイベント (システムコール) には、1 から 2047 までのイベント番号が付けられます。カーネルイベントのイベント名は `AUE_` と例えば、`creat()` システムコールのイベント番号は 4 でイベント名は `AUE_CREAT` です。

ユーザレベルのイベント

カーネルの外側でアプリケーションソフトウェアによって生成されるイベントの番号は、2048 から 65535 までの範囲にあります。イベント名は `AUE_` で始まり、その後イベントを表す小文字のニーモニックが続きます。ファイル `/etc/security/audit_event`、内で各イベントの番号を調べてください。表 2-1 に、ユーザに関連するイベントの一般的なカテゴリを示します。

表 2-1 監査イベントのカテゴリ

| 番号の範囲 | イベントのタイプ |
|-------------|------------------------|
| 2048-65535 | ユーザレベルの監査イベント |
| 2048-32767 | SunOS ユーザレベルのプログラム用に予約 |
| 32768-65536 | サードパーティアプリケーション用に使用可 |

監査レコード

各監査レコードには、監査された1つのイベントの発生が記述されます。誰がその動作を行ったか、どのファイルが影響を受けたか、どのような動作が試みられたか、いつ、どこでその動作が発生したかといった情報が含まれます。

監査イベントごとに保存される情報は、監査トークンのセットとして定義されます。1つのイベントに対して監査レコードが作成されるたびに、イベントの内容に従ってそのレコードに、トークンの一部またはすべてが書き込まれます。付録 A には、各イベントに定義された監査トークンと各トークンの意味がすべてリストされています。

監査レコードはトレール内に収集され (audit.log(4) のマニュアルページを参照)、`praudit` コマンドによってユーザが読める書式に変換できます (`praudit(1M)` のマニュアルページを参照)。詳細については、第 3 章を参照してください。

監査フラグ

監査フラグは監査対象となるイベントのクラスを示します。マシン全体で有効な監査関連のデフォルト値は、`audit_control` ファイル内のフラグによって各マシン上のすべてのユーザ用に指定されます。詳細については、12ページの「`audit_control` ファイル」を参照して下さい。

システム管理者は、監査フラグを `audit_user` ファイルにあるユーザエントリに入れることにより、各ユーザに対して監査を行う対象を修正できます。また、監査フ

ラグは、auditconfig コマンドの引数として使用できます (auditconfig(1M) のマニュアルページを参照)。

監査フラグの定義

表 2-2 に、事前に定義されている各監査クラスの監査フラグ (これはクラスを表す短縮名です)、ロング名、および説明を示します。システム管理者は、監査構成ファイル内の監査クラスを使用して、監査の対象となるイベントのクラスを指定します。audit_class ファイルを修正することにより、クラスの定義を追加したり、クラス名を変更できます (audit_class(4) のマニュアルページを参照)。

表 2-2 監査クラス

| 短縮名 | ロング名 | 短い説明 |
|-----|----------------|------------------------------------|
| no | no_class | イベントの事前選択をオフにするためのヌル値 |
| fr | file_read | データの読み取り、読み取りのためのオープンなど |
| fw | file_write | データの書き込み、書き込みのためのオープンなど |
| fa | file_attr_acc | オブジェクト属性へのアクセス: stat、pathconf など |
| fm | file_attr_mod | オブジェクト属性の変更: chown、flock など |
| fc | file_creation | オブジェクトの作成 |
| fd | file_deletion | オブジェクトの削除 |
| cl | file_close | close システムコール |
| pc | process | プロセスの操作: fork、exec、exit など |
| nt | network | ネットワークイベント: bind、connect、accept など |
| ip | ipc | System V の IPC 操作 |
| na | non_attrib | ユーザが原因ではないイベント |
| ad | administrative | 管理的な操作 |
| lo | login_logout | ログインとログアウトのイベント |

表 2-2 監査クラス 続く

| 短縮名 | ロング名 | 短い説明 |
|-----|-------------|-------------------|
| ap | application | アプリケーションが定義するイベント |
| io | ioctl | ioctl システムコール |
| ex | exec | プログラムの実行 |
| ot | other | その他 |
| all | all | 全フラグのセット |

監査フラグの構文

イベントのクラスは、付けられている接頭辞に応じて、成功したか失敗したかについて、成功した場合のみ、または、失敗した場合のみ、監査を行うことができます。監査フラグの構文は次のとおりです。

prefixflag

次の表に、成功した場合も失敗した場合も監査する接頭辞、成功した場合のみ、または失敗した場合のみ監査する接頭辞をそれぞれ示します。

表 2-3 監査フラグに使用する接頭辞

| 接頭辞 | 意味 |
|-----|------------------|
| なし | 成功と失敗の両方の場合に監査する |
| + | 成功の場合のみ監査する |
| - | 失敗の場合のみ監査する |

これらの監査フラグがどのように連動するかという例として、`lo` を考えてみます。これは「ログインとログアウトの成功した場合すべて、および、ログインの失敗した場合すべて」を意味する監査フラグです (ログアウトに失敗することはありません)

ん)。別の例としては、`-a11` は失敗したすべての種類の試みを意味し、`+a11` は成功したすべての種類の試みを意味します。



注意 - `a11` フラグは大量のデータを生成し、すぐに監査ファイルシステムをいっぱいにします。したがってこのフラグは、すべてを監査しなければならない特別な理由がある場合にのみ使用してください。

設定済みの監査フラグを変更する接頭辞

次の接頭辞を 3 つの方法のいずれかで使用します。つまり、すでに指定されているフラグを変更する場合に、`audit_control` ファイル内の `flags` 行を使用するか、`audit_user` ファイル内のユーザエントリの `flags` を使用するか、または `auditconfig` を使用して、監査フラグを変更します (`auditconfig(1M)` のマニュアルページを参照)。

次の表に示す接頭辞を監査クラスの短縮名とともに使用して、指定済みの監査クラスをオンまたはオフにします。これらの接頭辞は、指定済みのフラグのみをオンまたはオフにします。

表 2-4 指定済みの監査フラグを変更する接頭辞

| 接頭辞 | 意味 |
|-----------------|------------------------------|
| <code>^-</code> | 失敗した試みに対する監査をオフにする |
| <code>^+</code> | 成功した試みに対する監査をオフにする |
| <code>^</code> | 成功した試みと失敗した試みの両方に対して監査をオフにする |

`^-` 接頭辞は、次の例のように `audit_control` ファイルの `flags` 行で使用します。

次の例では、`lo` フラグと `ad` フラグが、すべてのログインと管理的な操作について成功と失敗の両方の場合に、監査することを指定しています。`-a11` は「失敗したすべてのイベント」を監査することを意味します。`^-` 接頭辞は「指定したクラスの、失敗した試みについての監査をオフにする」ため、`^-fc` フラグは、失敗したすべてのイベントの監査を指定した以前のフラグを変更します。したがって、この 2 つの

フィールドを合わせると、「ファイルシステムオブジェクトの作成に失敗した試みを除けば、失敗したイベントをすべて監査する」ことを意味します。

```
flags:lo,ad,-all,^-fc
```

audit_controlファイル

各マシン上の audit_control ファイルは、監査デーモンによって読み込まれます (audit_control(4) のマニュアルページを参照)。audit_control ファイルは /etc/security ディレクトリにあります。分散システムのマシンは監査ファイルシステムをさまざまな場所からマウントしているか、異なる順序で指定しているので、各マシン上では別の audit_control ファイルが管理されます。たとえば、マシン A の一次監査ファイルシステムは、マシン B の二次監査ファイルシステムになっている場合があります。

audit_control ファイル内では、4 種類の情報を 4 種類の行で指定します。

- 監査フラグ行 (flags:) には、マシン上のすべてのユーザに対して監査されるイベントクラスを定義する監査フラグが含まれます。ここで指定する監査フラグは、マシン全体の監査フラグまたはマシン全体の監査事前選択マスクと呼ばれます。監査フラグは、スペースなしでカンマで区切ります。
- 非帰因フラグ行 (naflags:) には、動作が特定のユーザに帰因しない場合に、監査されるイベントクラスを定義する監査フラグが含まれます。フラグはスペースなしでカンマで区切ります。
- 監査しきい値行 (minfree:) では、すべての監査ファイルシステムについて確保する最小空き領域のレベルを定義します。18ページの「監査デーモンが使用できるディレクトリ」を参照してください。minfree のパーセンテージは、0 以上でなければなりません。デフォルトは 20 パーセントです。
- ディレクトリ定義行 (dir:) では、監査トレールファイルを格納するためにマシンが使用する監査ファイルシステムとディレクトリを定義します。1行または複数のディレクトリ定義行を使用できます。auditd は、指定した順序でディレクトリ内の監査ファイルを開くので、dir: 行の順序は重要です (audit(1M) のマニュアルページを参照)。最初に指定される監査ディレクトリは、そのマシンの一次監査ディレクトリで、2つ目の監査ディレクトリは、二次監査ディレクトリです。監査デーモンは、最初の監査ディレクトリがいっぱいになると、監査トレールファイルを2つ目の監査ディレクトリに入れ、以下同様の処理を行います。

管理者は、構成時に `audit_control` ファイルをマシンごとに作成します。

管理者は、システム構成時に `audit_control` ファイルを作成した後で、それを編集できます。変更した後に `audit -s` を実行して監査デーモンに `audit_control` ファイルを再度読み込むように指示します。

注 - `audit -s` コマンドでは、既存のプロセスについて指定された事前選択マスクは変更されません。既存のプロセスについては `auditconfig`、`setaudit` (`getaudit(2)` のマニュアルページを参照)、または `auditon` を使用します。

audit_control ファイルのサンプル

次の例は、マシン `dopey` で使用する `audit_control` ファイルです。`dopey` では、監査サーバ `flinpen` 上にある 2 つの監査ファイルシステムと、別の監査サーバ `winken` からマウントした 3 つ目の監査ファイルシステムが使用されます。3 つ目の監査ファイルシステムは、`blinken` 上の監査ファイルシステムがいっぱいになった場合、または利用できない場合に使用されます。20 パーセントに指定された `minfree` は、ファイルシステムが 80 パーセントまで使用され、現在のマシンの監査データが、次に利用できる監査ディレクトリがあればそこに格納される場合に、警告スクリプト (`audit_warn(1M)` のマニュアルページを参照) を実行するように指示します。このフラグによって、すべてのログインと管理操作が(成功するかどうかにかかわらず) 監査されることと、ファイルシステムのオブジェクトの作成に失敗する場合を除き、すべてのタイプの失敗が監査されることが指定されます。

```
flags:lo,ad,-all,^-fc
naflags:lo,nt
minfree:20
dir:/etc/security/audit/blinken/files
dir:/etc/security/audit/blinken.1/files
#
# Audit filesystem used when blinken fills up
#
dir: /etc/security/audit/winken
```

audit_user ファイル内のユーザ監査フィールド

あるユーザを他のユーザと異なる方法で監査することが望ましい場合には、管理者は `audit_user` ファイルを編集してユーザに監査フラグを追加できます。これらの監査フラグが指定されている場合は、監査制御ファイル内で指定されているシステ

ム全体で有効なフラグと組み合わせられ、そのユーザに関してどのイベントクラスを監査するかが決定されます。管理者が `audit_user` ファイル内のユーザエントリに追加するフラグは、`audit_control` ファイルにあるデフォルトを次の2つの方法で変更します。1つは、このユーザについては決して監査されないイベントクラスのセットを指定する方法で、もう1つは、常に監査の対象となるイベントクラスのセットを指定する方法です。

`audit_user` ファイルの各ユーザエントリには、3つのフィールドがあります。最初のフィールドは `username` フィールド、2つ目は `always-audit` フィールド、3つ目は `never-audit` フィールドです。フィールドは順番に処理されるので、監査機能は2つ目の `always-audit` で使用可能になり、3つ目の `never-audit` で使用不可になります。

注 - `never-audit` フィールド内で `a11` を設定したままにするというのがよくある間違いです。この場合、そのユーザに関してすべての監査機能がオフになり、`never-audit` フィールド内で設定されたフラグセットが上書きされてしまいます。

あるユーザにフラグを使用することと、`always-audit` のセットからクラスを削除することとは異なります。たとえば、(次の例のように) ファイルシステムのオブジェクトの読み込みに成功した場合を除き、ユーザ `fred` の動作はすべて監査の対象にしたと仮定します(すべてのデータ読み込み動作を監査する場合に生成される監査データの4分の3程度に監査データを抑えながら、あるユーザに関するほとんどすべての動作を監査するには、この方法が適しています)。また、システムのデフォルトをユーザ `fred` に適用するとします。次に2つの `audit_user` エントリを示します。

正しいエントリ

```
fred:all,^+fr:
```

間違ったエントリ

```
fred:all:+fr
```

はじめの例は、「ファイルの読み込みが成功した場合を除き、常にすべての動作を監査する」ことを表しています。2つ目の例は、「常にすべての動作を監査するが、ファイルの読み込みに成功した場合は決して監査しない」ことを表しています。2つ目の例はシステムのデフォルト値が変更されるので、正しいエントリではありません。はじめの例では期待どおりの効果が達成されます。つまり、`audit_user` エントリで指定した内容の他に、初期に設定したデフォルト値が適用されます。

注・成功したイベントと失敗したイベントは別個に取り扱われるので、プロセスが生成する監査レコードの量は、そのイベントが成功したときよりも、失敗したときの方が多くなることがあります。

プロセスの監査特性

はじめてログインするときに次の監査特性が設定されます。

- プロセス事前選択マスク
- 監査 ID
- 監査セッション ID
- 端末セッション ID (ポート ID、マシン ID)

プロセス事前選択マスク

ユーザがログインするときには、login により、audit_control ファイルにあるマシン全体の監査フラグが audit_user ファイルにあるユーザ固有の監査フラグ (もしあれば) と組み合わせられ、そのユーザのプロセスに使用するプロセス事前選択マスクが確立されます。プロセス事前選択マスクは、各監査イベントクラス内のイベントで監査レコードを生成させるかどうかを指定します。

プロセス事前選択マスクを取得するアルゴリズムは次のとおりです。つまり、audit_control ファイル内の flags: 行にある監査フラグが、audit_user ファイル内のユーザエントリの always-audit フィールドにあるフラグに追加されます。audit_user ファイル内のユーザエントリの never-audit フィールドは、このときに合計から差し引かれます。

ユーザのプロセス事前選択マスク
= (flags: 行 + always audit フラグ) - never audit フラグ

監査 ID

ユーザがログインすると、プロセスはまた、ユーザの監査 ID を取得し、この監査 ID はユーザの初期プロセスが起動するすべての子プロセスに継承されます。監査

ID はアカウントの追跡を強制するのにも役立ちます。ユーザが root になった後も、監査 ID は変わらずそのまま残ります。各監査レコード内に保存された監査 ID を使用すると、管理者はいつでも動作を追跡してログインしたもとのユーザまでたどることができます。

監査セッション ID

監査セッション ID はログイン時に割り当てられ、すべての子孫のプロセスに継承されます。

端末 ID

端末 ID は、ホスト名とインターネットアドレスからなり、その後にユーザがログインした物理デバイスを識別する固有の番号が続きます。通常、ログインはコンソールから行われ、そのコンソールデバイスに対応する番号は 0 になります。

監査トレールの作成

監査トレールは監査デーモンによって作成されます (詳細は、auditd(1M) のマニュアルページを参照してください)。監査デーモンは、マシンが起動されるとそのマシン上で起動されます。auditd は、ブート時に起動されると、監査トレールデータを収集し、監査レコードを監査ファイルに書き込む処理を受け持ちます。このファイルを監査ログファイルとも呼びます。ファイルの書式についての詳細は、audit.log(4) のマニュアルページを参照してください。

監査デーモンは root として動作します。監査デーモンによって作成されるファイルは、すべて root に所有されます。auditd は、監査するクラスがなくても動作を継続し、監査レコードを置く場所を探します。また、カーネルの監査バッファがいっぱいになったためにマシンの他の動作が中断されても、auditd は動作を続けます。監査動作を続行できるのは、auditd が監査の対象ではないからです。

監査デーモンは、一度に 1 つしか実行できません。第 2 の監査デーモンを起動しようとする、エラーメッセージが表示され、その新しい監査デーモンが終了します。監査デーモンに問題がある場合は、audit -t を使用して auditd を正常に終了させ、手作業で再起動してみる必要があります。

デーモンが監査ディレクトリを切り替えたり、問題(記憶領域不足など)が発生すると、auditdによってaudit_warnスクリプトが実行されます。分散システムであることから、audit_warnスクリプトはaudit_warnの別名にメールを送り、コンソールにメッセージを送ります。サイトでは、ニーズに合わせてaudit_warnをカスタマイズする必要があります。audit_warnスクリプトをカスタマイズする方法については、19ページの「audit_warnスクリプト」を参照してください。

audit_data ファイル

auditdは、各マシン上で起動されると、ファイル/etc/security/audit_dataを作成します。このファイルの書式は、1つのエントリに、コロんで区切られた2つのフィールドがあります(audit_data(4)のマニュアルページを参照)。第1のフィールドは監査デーモンのプロセスIDで、第2のフィールドは、監査デーモンが監査レコードを現在書き込んでいる監査ファイルのパス名です。

```
# cat /etc/security/audit_data
116:/etc/security/audit/blinken.1/files/19910320100002.not_terminated.lazy
```

監査デーモンの役割

次に、監査デーモンauditdの役割を示します。

- auditdは、audit_controlファイル内で指定されたディレクトリ内の監査ログファイルを、指定された順序で開き、閉じます。
- auditdは、監査データをカーネルから読み取り、監査ファイルに書き込みます。
- auditdは、監査ディレクトリ内のデータ量がaudit_controlファイル内で指定された上限を超えると、audit_warnスクリプトを実行します。デフォルトでは、このスクリプトはaudit_warnの別名とコンソールに警告を送ります。
- システムのデフォルト構成では、すべての監査ディレクトリがいっぱいになると、監査レコードを生成するプロセスが中断されます。また、auditdはコンソールとaudit_warnの別名にメッセージを書き込みます(autoconfigを使用すると監査方針を再構成できます)。この時点では、システム管理者だけがログインして、監査ファイルをテープに書き込むか、システムから監査ファイルを削除するか、または他のクリーンアップ処理を実行できます。

マシンがマルチユーザモードで起動されるときに監査デーモンが起動されたり、または、監査デーモンがaudit -s コマンドによりauditdファイルを編集後にもう一度読み取るように命令を受けると、auditdは必要な空き容量を判断し、audit_controlファイルからディレクトリのリストを読み取り、それを監査ファイルの作成場所の候補として使用します。

監査デーモンは、このディレクトリのリストへのポインタを最初から管理します。監査デーモンが監査ファイルを作成しなければならなくなるたびに、監査デーモンはそのカレントポインタから始めて、監査ファイルをリスト内の最初の使用可能ディレクトリに入れます。管理者が `audit -s` コマンドを入力すると、このポインタをリストの先頭にリセットできます。`audit -n` コマンドを使用して、新しい監査ファイルに切り替えるようにデーモンに命令すると、新しいファイルは現在のファイルと同じディレクトリ内で作成されます。

監査デーモンが使用できるディレクトリ

監査デーモンがあるディレクトリを使用するには、そのディレクトリにアクセスできることが条件となります。つまり、そのディレクトリがマウントされ、ネットワーク経由 (リモートの場合) のアクセスが許可され、ディレクトリに対するアクセス権がなければなりません。また、監査ファイルをディレクトリに保存するには、十分な空き領域がなければなりません。`audit_control` ファイルの `minfree:` 行を編集して、デフォルトの 20 パーセントを変更できます。`minfree` の設定が 20 パーセントというデフォルトの最小空き領域の場合は、ファイルシステムが 80 パーセントを超えていっぱいになると、`audit_warn` の別名に電子メール通知が送信されます。

十分な空き領域が残っているディレクトリがリストになれば、デーモンはリストの先頭からはじめて最後に達するまで、ハード限界に達せず使用可能領域が残っている最初のアクセス可能なディレクトリを選択します。デフォルト構成では、適切なディレクトリがなければ、デーモンは監査レコードの処理を停止し、監査レコードを生成中のすべてのプロセスが中断されるまで、カーネル内に蓄積します。

監査ファイルを常に管理可能な状態に保つ

監査ファイルを管理可能なサイズに保つために、監査ファイルを定期的に切り替える `cron` ジョブを設定できます (`cron(1M)` のマニュアルページを参照)。切り替え間隔は、収集される監査データの量に応じて、1 時間ごとから 1 日に 2 度までの範囲で設定できます。データにフィルタをかけ、不要な情報を削除して圧縮できます。

audit_warn スクリプト

監査デーモンは、監査レコードの書き込み中に異常な条件が発生すると、`/etc/security/audit_warn` スクリプトを起動します。audit_warn(1M)のマニュアルページを参照してください。このスクリプトをサイトでカスタマイズして、手作業による介入が必要な場合に警告を出したり、自動処理を行わせることができます。どのエラー条件が発生した場合も、audit_warn はメッセージをコンソールに書き込み、audit_warn の別名に送ります。管理者は、BSM を使用可能にした後で、この別名を設定しておく必要があります。

監査デーモンは次の条件を検出すると audit_warn を起動します。

- 監査ディレクトリが minfree の許容値を超えていっぱいになった場合 (minfree、つまりソフト限界は、監査ファイルシステム上で使用可能な領域のパーセンテージです)。

audit_warn スクリプトは、文字列 `soft` と、使用可能領域が下限を下回ったディレクトリの名前を使用して起動されます。監査デーモンは、次に適切なディレクトリに自動的に切り替えて、この新しいディレクトリが minfree の上限に達するまで監査ファイルに書き込みます。監査デーモンは、audit_control にリストされた順序で残りの各ディレクトリにアクセスし、それぞれが minfree の制限に達するまで監査レコードを書き込みます。

- すべての監査ディレクトリが minfree のしきい値を超えて満たされた場合。

引数として文字列 `allsoft` を使用して、audit_warn スクリプトが起動されます。コンソールにメッセージが書き込まれ、audit_warn の別名にメールが送られます。

audit_control 内にリストされたすべての監査ディレクトリがそれぞれの minfree 制限に達すると、監査デーモンは最初の監査ディレクトリに戻って、そのディレクトリが完全にいっぱいになるまで監査レコードを書き込みます。

- 監査ディレクトリが完全にいっぱいになり、残りの領域がなくなった場合。

引数として文字列 `hard` とディレクトリ名を使用して、audit_warn スクリプトが起動されます。コンソールにメッセージが書き込まれ、audit_warn の別名にメールが送られます。

監査デーモンは、使用可能領域が残っている次の適切なディレクトリがあれば、それに自動的に切り替えます。その後は、audit_control 内でリストされた順

番に、残りの各ディレクトリにアクセスし、それぞれがいっぱいになるまで監査レコードを書き込みます。

- すべての監査ディレクトリが完全にいっぱいになった場合。引数として文字列 `allhard` を使用して、`audit_warn` スクリプトが起動されます。

デフォルト構成では、コンソールにメッセージが書き込まれ、`audit_warn` の別名にメールが送られます。監査レコードを生成中のプロセスは中断されます。監査デーモンはループに入り、領域が使用可能になるのを待って監査レコードの処理を再開します。監査レコードが処理されないうちは、監査対象の動作も発生しません。監査レコードを生成しようとするプロセスはすべて中断されます。このため、別の監査管理アカウントを設定し、監査機能を使用可能にせずに操作できるようにしておくことが推奨されます。この方法を使用すると、管理者は中断せずに操作を続けることができます。

- 内部エラーが発生した場合。つまり、別の監査デーモンプロセスがすでに実行されている場合 (文字列 `ebusy`)、一時ファイルを使用できない場合 (文字列 `tmpfile`)、`auditsvc()` システムコールが失敗した場合 (文字列 `auditsvc`)、または監査のシャットダウン中に信号を受信した場合 (文字列 `postsigterm`)。

`audit_warn` の別名にメールが送られます。

- `audit_control` ファイルの内容にエラーが見つかった場合。デフォルトでは、`audit_warn` の別名にメールが送られ、コンソールにメッセージが送られません。

auditreduce コマンドの使用法

`auditreduce` を使用すると、1つまたは複数の入力監査ファイルから監査レコードをマージしてまとめたり、監査レコードの事後選択を実行できます。 `auditreduce(1M)` のマニュアルページを参照してください。監査トレール全体をマージするには、システム管理者は分散システムのすべての監査ファイルシステムがマウントされているマシン上で、`auditreduce` コマンドを入力します。

BSM を実行する複数のマシンが分散システムの一部として管理されているときは、各マシンが監査可能なイベントを実行し、監査レコードをマシン固有の監査ファイルに書き込みます。この手順によってソフトウェアが単純化され、マシンが障害を起こした場合の信頼性が高まります。しかし、各マシンによって独自の監査ファイルセットが生成されるので、`auditreduce` を使用しないと、すべてのファイルを個別に調べてどのユーザーに帰因するかを判断しなければなりません。

`auditreduce` コマンドによって、監査トレール全体の管理作業を効率化できます。`auditreduce` (または、より高いインタフェースを提供するために独自に作成したシェルスクリプト) を使用すると、レコードの生成方法や格納場所に関係なく、システム内のすべてのファイルの論理上の組み合わせを1つの監査トレールとして読むことができます。

`auditreduce` プログラムは、監査デーモンによって生成された監査トレールを処理します。1つまたは複数の監査ファイルからレコードが選択され、マージされて、1つの時系列順のファイルが生成されます。`auditreduce` のマージ機能と選択機能は論理的に他に依存しません。`auditreduce` はレコードが読み取られると、入力ファイルがマージされてディスクに書き込まれる前に、そこからメッセージを選択します。

オプションを指定しなければ、`auditreduce` は監査トレール全体 (デフォルトの監査ルートディレクトリ `/etc/security/audit` 内のすべてのサブディレクトリ内のすべての監査ファイルです) をマージして、すべての監査レコードを標準出力に送ります。`praudit` コマンドによって、レコードをユーザが読める書式に変換されます。

次の例は、`auditreduce` コマンドの一部のオプションによって実行される動作です。

- 特定の監査フラグによって生成される監査レコードだけが出力に含まれるように設定できます。
- 特定の1人のユーザによって作成される監査レコードを要求できます。
- 特定の日付に生成された監査レコードを要求できます。

引数を指定しなければ、`auditreduce` はデフォルトの監査ルートディレクトリである `/etc/security/audit` の下のすべてのサブディレクトリ内で、`date.date.hostname` ファイルが入っている `files` ディレクトリを検索します。`auditreduce` コマンドは、さまざまなホストの監査データ (図 2-1) や、さまざまな監査サーバの監査データ (図 2-2) が別々のディレクトリに入っている場合に便利です。

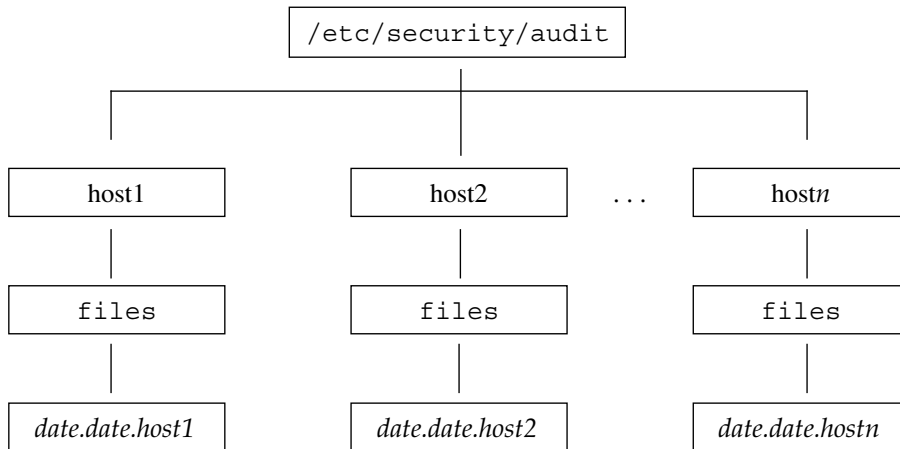


図 2-1 ホスト別に分類された監査トレール

監査データはデフォルトディレクトリに入っていないかまいません。これは、`/etc/security/audit` のパーティションが小さすぎるため、または、別のパーティションを `/etc/security/audit` にシンボリックリンクせずに、そのパーティションに監査データを保存したい場合です。`auditreduce` に `/etc/security/audit` の代わりに別のディレクトリ (`-R`) を指定するか、または特定のサブディレクトリ (`-S`) を指定できます。

```

# auditreduce -R /var/audit-alt
# auditreduce -S /var/audit-alt/host1
  
```

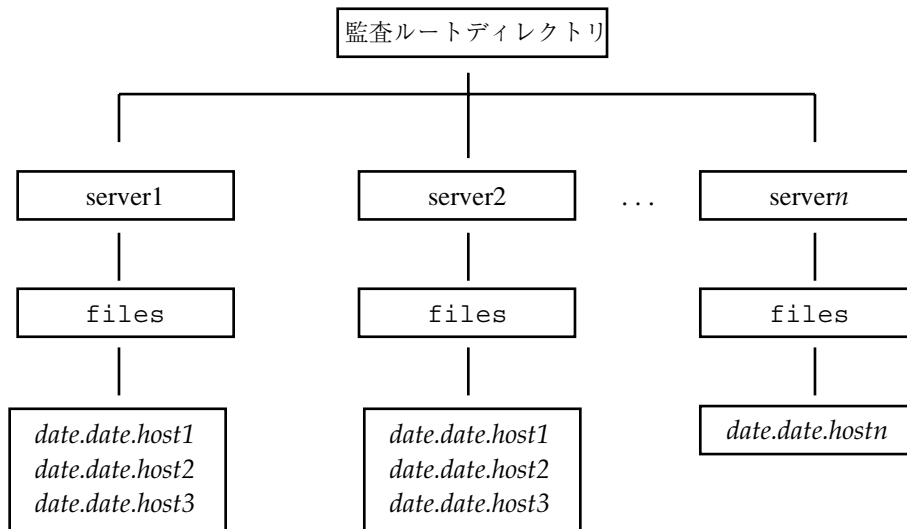


図 2-2 サーバ別に分類された監査トレール

次のように特定のファイルをコマンド引数として指定すると、それらのファイルのみを処理するように `auditreduce` に対して指示できます。

```
# auditreduce /var/audit/bongos/files/1993*.1993*.bongos
```

他のオプションとコマンドの使用例については、`auditreduce (1M)` のマニュアルページを参照してください。

監査コストの制御

監査処理によってシステム資源が消費されるので、どの程度詳しく記録するかを制御しなければなりません。監査の対象を決めるときには、監査に伴う次の3つのコストを考慮してください。

- 処理時間の増大に伴うコスト
- 監査データの分析に伴うコスト
- 監査データの格納に伴うコスト

処理時間の増大に伴うコスト

処理時間の増大に伴うコストは、監査に関連する3つのコストの中では最も重要性の低い問題です。第1に、通常は、イメージ処理や複雑な計算処理など、計算集中型のタスクを実行中には監査処理が発生しないからです。第2に、1人のユーザだけが使うワークステーションではCPUに十分な余裕があるからです。

分析に伴うコスト

分析に伴うコストは、収集される監査データの量にほぼ比例します。この種のコストには、監査レコードをマージして検討する所要時間や、それをファイルに保存して安全な場所に保管する所要時間が含まれます。

生成するレコードが少ないほど分析に要する時間も短くなるので、この後の節ではサイトのセキュリティ目標を保ちつつ、収集するデータ量を削減する方法について説明します。

格納に伴うコスト

格納コストは、監査に伴うコストのうちで最も重要です。監査データの量は次の要素によって左右されます。

- ユーザ数
- マシン数
- 使用量
- 必要なセキュリティの程度

各要素は状況により異なるので、監査データの格納用に前もって確保しておくディスク容量を決定できるような計算式はありません。

完全な監査 (all フラグで指定します) の場合は、ディスクがすぐいっぱいになってしまいます。中程度のサイズのプログラム (5 つのファイルで合計 5000 行のプログラムなど) をコンパイルするなど、1 分もかからないごく単純なタスクでも、数千の監査レコードが生成され、何メガバイトものディスク領域を消費する可能性があります。したがって、事前選択機能を使用して、生成されるレコードの量を削減しておくことが大変重要になります。たとえば、すべてのクラスを監査するのではなく、fr クラスを省略すると、監査ボリュームを 3 分の 2 以上も削減できます。また、監査レコードが作成された後も、必要なディスク容量を削減するために監査ファイルを効率よく管理することが重要です。

この後の節では、サイトのセキュリティに関するニーズを満たしつつ、監査対象を選択して収集される監査データの量を削減し、格納に伴うコストを削減する方法について、いくつかのヒントを示します。また、監査ファイル記憶装置と保存手順を設定し、記憶領域の所要量を削減する方法についても説明します。

監査機能を構成する前に、監査フラグと、フラグが立てられるイベントのタイプを理解しておく必要があります。サイトで必要なセキュリティの程度と、管理の対象となるユーザのタイプに基づいて、組織の監査上の基本方針を設定してください。

プロセス監査の事前選択マスクが動的に変更されない限り、ユーザのログイン時に設定される監査特性は、ログインセッション中に発生するすべてのプロセスに継承されます。また、データベースが変更されない限り、プロセス事前選択マスクは後続のすべてのログインセッションに適用されます。

動的制御とは、プロセスの実行中に管理者が設定する制御のことです。この制御が有効なのは、影響を受けるプロセス (および、その子プロセス) が存在する間だけです。次のログイン時には有効になりません。監査コマンドはログインしている現在のマシンにのみ適用されるので、動的制御は一度に 1 つのマシンにのみ適用さ

れます。ただし、あるマシン上で動的な変更を行う場合は、それと同時にすべてのマシン上でも同じように変更する必要があります。

各プロセスは、監査クラス用に 1 ビットのフラグを 2 組ずつ持っています。1 組目はそのクラスのイベントが正常に要求されたときにプロセスが監査対象になるかどうかを制御します。2 組目は、イベントが要求されたが (何らかの原因で) 失敗したときに、プロセスが監査対象になるかどうかを制御します。これは、システムのセキュリティをやぶろうとする場合のブラウズ操作やその他の試行操作を検出するために使用されるため、一般的にプロセスが成功した場合よりも失敗した場合の方が詳しく監査されます。

静的なデータベース内でユーザ単位の監査制御情報を提供するだけでなく、1 台のマシン上でユーザのプロセスが動作している間に、監査の状態を動的に調整できます。

特定のユーザについての監査フラグを変更するには、`-setpmask`、`-setumask`、または `-setsmask` オプションを指定して `auditconfig` コマンドを使用します。このコマンドによって、1 つのプロセス、監査セッション ID、または監査ユーザ ID のプロセス監査フラグがそれぞれ変更されます。`auditconfig(1M)` のマニュアルページと 38 ページの「`auditconfig` コマンド」を参照してください。

通常の利用者の監査

管理者は、デフォルト構成に合わせて監査機能を設定します。`audit_control` ファイル内で指定したシステムワイドな監査フラグに従って、すべてのユーザと管理者を管理対象にすることができます。個々のユーザに対する監査機能を微調整するには、`audit_user` ファイル内でそのユーザのエントリを変更します。`audit_control(4)` と `audit_user(4)` のマニュアルページを参照してください。また、新しくユーザを追加するときに、ユーザのエントリに監査フラグを追加できますが、新規ユーザのアカウントのロックを解除し、そのユーザのセキュリティ属性を構成した直後に、そのユーザの監査機能を設定する必要があります。

効率的な監査

この節で説明する方法により、組織のセキュリティ目標を達成する一方で、監査効率を高めることができます。

- 一定の割合のユーザのみを任意の時間にランダムに監査する。
- 監査データをリアルタイムで監視して異常な動作がないかどうかを調べる (特定の動作に関して生成される監査トレールを監視し、疑わしいイベントが発生した場合には特定のユーザまたはマシンの監査レベルを上げるための手順を設定します)。
- 監査ファイルを組み合わせ、縮小し、圧縮してディスクの所要量を削減し、オフラインで格納する手順を設定する。

もう1つの方法は、監査トレールをリアルタイムで監視することです。異常なイベントが検出された場合に、それに応じて特定のユーザまたはマシンの監査レベルを自動的に上げるようなスクリプトを作成できます。

監査トレールをリアルタイムで監視し、異常なイベントが発生していないかどうかを調べるには、すべての監査ファイルサーバ上で監査ファイルの生成を監視し、それを `tail` コマンド (`tail(1)` のマニュアルページを参照) で処理するスクリプトを作成します。`tail -of` の出力をパイプにより `praudit` に渡すようにすると、監査レコードが生成されたときすぐにそれを表示して確認できます。この出力を分析して、異常なメッセージのタイプや他の兆候の有無を調べ、監査担当者に送付したり、自動応答の切り替えに使用できます。このスクリプトを作成して、アクティブな新しい `not_terminated` 監査ファイルがないかどうか、また、ファイルが書き込まれなくなったとき (つまり、新しいファイルに切り替えられたとき) に `tail` プロセスが終了しているかどうかを、監査ディレクトリ内で絶えず調べるようにする必要があります。

▼ audit ファイルを組み合わせる方法

- ◆ `-O` オプションを指定して `auditreduce` を使用すると、複数の監査ファイルを組み合わせることで1つにまとめ、指定した出力ファイルに保存できます。

`auditreduce` を使用すると、組み合わせと削除を自動的に実行できますが (`auditreduce(1M)` のマニュアルページの `-C`、`-D` オプションを参照)、通常はファイルを手作業で (`find` を使用して) 選択し、指定したファイルセットを `auditreduce` を使用して組み合わせる方が簡単です。この方法で `auditreduce` を使用すると、入力ファイル内のすべてのレコードがマージされて、1つの出力ファイルにまとめられます。その後、入力ファイルを削除し、`auditreduce` で検索できるように、出力ファイルをディレクトリ `/etc/security/audit/server-name/files` に保存する必要があります。

```
# auditreduce -O combined-filename
```

また、auditreduce プログラムでは、入力ファイルを組み合わせるときに不要なレコードを除去して、出力ファイル内のレコード数を削減できます。完全な監査トレールを検索する必要がある場合にはバックアップテープから回復できることを前提にして、ログインとログアウトイベントを除いた、数カ月のすべてのレコードを auditreduce を使用して削除するようなこともあります。

```
# auditreduce -O daily.summary -b 19930513 -c lo; compress *daily.summary
# mv *daily.summary /etc/security/summary.dir
```

監査トレールについて

この節では、監査ファイルの格納場所、命名方法、分散システム全体で監査ファイルの記憶領域を管理する方法について説明します。

監査トレールは、監査デーモン auditd が起動されるときに作成され、新しい監査ファイルが作成されたとき、または監査デーモンが終了するときに閉じられます。監査トレールは複数の監査ディレクトリ内の監査ファイルから構成される場合や、監査ディレクトリに複数の監査トレールが入っている場合があります。

ほとんどの場合監査ディレクトリは、別個の監査ファイルシステムパーティションになります。他のファイルシステムに組み込むこともできますが、この方法はお勧めできません。

原則として、一次監査ディレクトリは別のパーティションにマウントされた専用の監査ファイルシステム内に配置します。通常、すべての監査ファイルシステムは /etc/security/audit のサブディレクトリです。これらのファイルシステムは、監査ディレクトリが監査ファイルでいっぱいになっても、パーティションをそのまま通常どおり使用できるように、専用の監査ファイルシステムにする必要があります。

監査ディレクトリは、監査専用でない他のファイルシステム内に物理的に配置することもできますが、最後の手段として確保しているディレクトリを除き、この方法は使用しないでください。最後の手段として確保しているディレクトリとは、他の適切なディレクトリが使用できないときに限り監査ファイルが書き込まれるディレクトリです。また、専用の監査ファイルシステムの外側に監査ディレクトリを配置できるのは、監査機能がオプションで、監査トレールを保存することよりもディスクをフル活用することの方が重要なソフトウェア開発環境などです。セキュリティが重視される実行環境では、監査ディレクトリを他のファイルシステム内に入れることは許されません。

ディスクフルマシンには、少なくとも 1 つはローカルの監査ディレクトリを用意する必要があります。このディレクトリは、監査サーバと通信できない場合に、最後の手段として確保しているディレクトリとして使用できます。

監査ディレクトリは、読み取り/書き込み (*rw*) オプションを使用してマウントしてください。監査ディレクトリをリモートで (*NFS* を使用して) マウントするときは、*intr* オプションも使用してください。監査ファイルシステムを、格納先の監査サーバ上でリストしてください。エクスポートリストには、監査サーバを使うよう構成されたすべてのマシンが含まれるはずですが、

監査ファイルシステムを、格納先の監査サーバ上でリストしてください。エクスポートリストには、監査サーバを使うよう構成されたすべてのマシンが含まれるはずですが、

監査ファイルの詳細

各監査ファイルは、それ自身で意味がわかるレコードの集合です。ファイル名には、レコードが生成された時間の範囲と、それを生成したマシン名が含まれます。

監査ファイルの命名

完全な監査ファイルには、次の書式の名前が付いています。

```
start-time.finish-time.machine
```

この場合、*start-time* は監査ファイル内の最初の監査レコードの生成時刻、*finish-time* は最後のレコードの生成時刻、*machine* はファイルを生成したマシンの名前です。監査ファイルの名前の例については、29ページの「閉じられた監査ファイルの名前の例」を参照してください。

監査ログファイルがまだアクティブな場合は、次の書式の名前が付いています。

```
start-time.not_terminated.machine
```

監査ファイル名の使用方法

auditreduce は、ファイル名のタイムスタンプを使用して、要求された特定期間内のレコードが入ったファイルを検索します。このタイムスタンプが重要なのは、1

カ月分以上の監査ファイルがオンライン上に存在する可能性があり、24 時間以内に生成されたレコードをすべて検索するとコストが大きくなりすぎるからです。

タイムスタンプの書式と意味

start-time と *end-time* は 1 秒単位のタイムスタンプで、グリニッジ標準時で指定されます。その書式は、次のように年が 4 桁で、月、日、時、分、秒が 2 桁ずつになっています。

```
YYYYMMDDHHMMSS
```

このタイムスタンプにはグリニッジ標準時が使用されるので、夏時間によるずれがあっても正しい順序でソートされることが保証されます。また、グリニッジ標準時が使用されるので、日時を把握しやすいように現在の時間帯に変換しなければなりません。監査ファイルを `auditreduce` ではなく標準ファイルコマンドで操作するときには、この点に注意してください。

まだアクティブなファイルのファイル名の例

まだアクティブなファイル名の書式は次のとおりです。

```
YYYYMMDDHHMMSS.not_terminated.hostname
```

次の例を参照してください。

```
19900327225243.not_terminated.lazy
```

監査ログファイルの名前には開始日が使用されるので、上記の例はグリニッジ標準時の 1990 年 3 月 27 日午後 10:52:43 から始まったことがわかります。ファイル名のうち `not_terminated` は、このファイルがまだアクティブであるか、または `auditd` が予期しないときに割り込まれたことを意味します。末尾の名前 `lazy` は、監査データが収集されているホストの名前です。

閉じられた監査ファイルの名前の例

閉じられた監査ログファイルの名前の書式は次のとおりです。

```
YYYYMMDDHHMMSS.YYYYMMDDHHMMSS.hostname
```

次の例を参照してください。

```
19900320005243.19900327225351.lazy
```

上記の例は、グリニッジ標準時の 1990 年 3 月 20 日の午前 12:52:43 に始まったことがわかります。このファイルは、グリニッジ標準時の 3 月 27 日午後 0:53:51 に閉じられました。末尾の名前 lazy は、監査データが収集されているマシンのホスト名です。

auditd が予期しないときに割り込まれるといつも、その時点で開いていた監査ファイルは not_terminated で終わるファイル名タイムスタンプを取得します。また、マシンがリモートでマウントされた監査ファイルに書き込んでいるときに、ファイルサーバがクラッシュするか、またはアクセスできなくなると、not_terminated で終わるタイムスタンプがカレントファイルの名前に付いたままになります。監査デーモンは、古い名前の監査ファイルをそのまま残して新しい監査ファイルを開きます。

not_terminated マークが付いたアクティブでないファイルの処理

auditreduce コマンドは、not_terminated マークが付いたファイルを処理しますが、この種のファイルの終わりには不完全なレコードが入っていることがあるので、それ以上処理するとエラーが発生する可能性があります。エラーを防ぐには、不完全なレコードが入っているファイルを整理してください。ファイルを整理する前に、そのファイルに auditd が書き込み中でないかどうかを確認してください。チェックするには、audit_data ファイルを調べて auditd の現在のプロセス番号を確認します。そのプロセスがまだ実行中で、audit_data 内のファイル名が問題のファイルと同じ場合は、そのファイルを整理しないでください。

auditreduce で -O オプションを使用するとファイルを整理できます。これによって、古いファイル内のすべてのレコードが入った新しいファイルが作成され、正しいファイル名タイムスタンプが付けられます。この処理を実行すると、各監査ファイルの先頭に付いていた以前のファイルポインタは失われます。

または、ファイル全体を読み取り、最後のレコードを突き止めてファイル名を変え、不完全なレコードを整理するプログラムを作成することもできます。また、以前のファイルポインタをそのまま残し、次にどのファイルを使用するかを決定するプログラムも作成できます。

▼ 監査パーティションを作成してエクスポートする方法

1. 各マシンに少なくとも 1 つの「一次監査ディレクトリ」を割り当てます。
一次監査ディレクトリは、通常の条件下でマシンが監査ファイルを配置するディレクトリです。
2. 一次監査ディレクトリとは異なる監査ファイルサーバ上で、各マシンに少なくとも 1 つの「二次監査ディレクトリ」を割り当てます。
二次監査ディレクトリは、一次ディレクトリがいっぱいになった場合や、ネットワーク障害、NFS サーバのクラッシュなどの原因でアクセスできなくなった場合に、マシンが監査ファイルを配置するディレクトリです。
3. 各ディスクフルマシン上で、最後の手段として確保するローカルの監査ディレクトリ (できれば専用の監査ファイルシステム) を作成します。このディレクトリは、ネットワークにアクセスできなくなったときや、一次と二次のディレクトリが使用できないときに使用されます。
4. 一次と二次の宛先として使用するディレクトリを、システム内の監査サーバに均等に分散させます。
5. この節で説明した要件に従って監査ファイルシステムを作成します。

`/etc/security` ディレクトリには、すべての監査ファイルと、監査制御に関連する他のファイルがいくつか入ったサブディレクトリがあります。`/etc/security` ディレクトリにはマシンごとの `audit_data` ファイルが入っており、ブート時に監査デーモンを正常に起動するには、このファイルが使用可能でなければならないので、`/etc/security` ディレクトリがルートファイルシステムになければなりません。

監査用の事後選択ツールは、デフォルトで `/etc/security/audit` の下のディレクトリ内を検索します。このため、監査サーバ上の最初の監査ファイルシステムのマウントポイントに使用するパス名は、`/etc/security/audit/` です (この場合、`server-name` は監査サーバ名です)。監査サーバ上に複数の監査パーティションがある場合、2 番目のマウントポイントの名前は `/etc/security/audit/server-name.1`、3 番目の名前は `/etc/security/audit/server-name.2` というようになります。

たとえば、監査サーバ `winken` 上で使用可能な監査ファイルシステムの名前は、`/etc/security/audit/winken` と `/etc/security/audit/winken.1` になります。

この監査サーバ上では、各監査ファイルシステムに `files` というサブディレクトリもなければなりません。この `files` サブディレクトリに監査ファイルが格納され、`auditreduce` コマンドによって検索されます。たとえば、監査サーバ `winken` 上の監査ファイルシステムには `files` サブディレクトリがあり、その完全パス名は `/etc/security/audit/winken/files` となります。

各マシン上のローカルの `audit_control` ファイルが、監査ファイルに対して `files` サブディレクトリに監査ファイルを格納するように指示しているかどうかを確認する必要があります。次の例は、`eagle` から監査ファイルシステムをマウントしているマシン上の `audit_control` ファイルの `dir:` 行を示しています。

```
dir: /etc/security/audit/eagle/files
```

監査サーバ上で (何らかの理由で) `/etc/security/audit/server-name [suffix]` ディレクトリが使用できないときに、マシンのローカルルートファイルシステムが監査ファイルでいっぱいになるのを防ぐには、別の階層レベルが必要です。監査サーバ上には `files` サブディレクトリがあり、どのクライアントにも `files` サブディレクトリがないため、マウントに失敗した場合に監査ファイルをローカルのマウントポイントディレクトリ上で自動的に作成できなくなります。

各監査ディレクトリに監査ファイル以外のデータが入っていないかどうかを確認してください。

6. 監査ファイルシステムに必要な許可を割り当てます。

下記の表 2-5 は、`/etc/security/audit/server-name files` ディレクトリと、その下に必ず作成する `files` ディレクトリ上になければならない許可を示しています。

表 2-5 監査ファイルの許可

| 所有者 | グループ | 許可 |
|------|-------|------|
| root | staff | 2750 |

audit_control ファイルのエントリの例

audit_control ファイルに dir: エントリを追加するときは、files サブディレクトリまでの完全パスを指定しなければなりません。次の例は、監査ファイルが専用のローカルディスクに格納されるサーバ blinken に関する audit_control ファイルの dir: エントリを示しています。

```
# cat /etc/security/audit_control
dir:/etc/security/audit/blinken.1/files
dir:/etc/security/audit/blinken.2/files
```

▼ 監査を構成する方法

次の手順は、監査ディレクトリを設定し、監査の対象となる監査クラスを指定するために必要な操作の概要を示しています。

1. ディスクをフォーマットし、パーティションに分割して、専用の監査パーティションを作成します。

経験では、分散システム上に配置するマシンごとに 100 M バイトを割り当てることになっています。ただし、どれくらいのディスク容量が自分のサイトに必要かは、どの程度の監査を実施するかによって異なり、マシン 1 台当たり 100 M バイトをはるかに上回るようなこともあります。

2. 監査ファイルシステムを専用パーティションに割り当てます。

NFS でマウントされた監査ファイルシステムを使用できない場合に備えて、ディスクフルマシンごとにローカルマシン上でバックアップ監査ディレクトリを作成しておく必要があります。

3. 各マシンがシングルユーザモードになっている間に、各専用監査パーティション上で `tunefs -m 0` を実行して、予約済みのファイルシステム領域を 0 パーセントに縮小します。

予約領域のパーセンテージ (minfree 制限と呼びます) は、audit_control ファイル内で監査パーティションに関して指定されています。デフォルトは 20 パーセントですが、このパーセンテージは調整できます。この値はサイトごとに audit_control ファイル内で設定するので、すべてのファイルシステム用にデフォルトで自動的に予約され確保されているファイルシステム領域を削除する必要があります。

4. 監査サーバ上で各監査ディレクトリに必要な許可を設定し、各監査ディレクトリ内で `files` というサブディレクトリを作成します。
 `chown` と `chmod` を使用して、各監査ディレクトリと各 `files` サブディレクトリに必要な許可を割り当てます。
5. 監査サーバを使用している場合は、`/etc/dfs/dfstab` ファイルといっしょに監査ディレクトリをエクスポートします。
6. `files` サブディレクトリを指定して、各マシン上の `audit_control` ファイル内にすべての監査ディレクトリに関する `audit_control` ファイルエントリを作成します。
7. 各監査クライアント上で、`/etc/vfstab` ファイル内に監査ファイルシステムに関するエントリを作成します。
8. 各監査クライアント上で、マウントポイントのディレクトリを作成し、`chmod` と `chown` を使用して適切な許可を設定します。

▼ 監査構成の計画を立てる方法

まず、監査トレール記憶領域の計画を作成します。

1. `/etc/security/audit_class` ファイル内で、サイトで必要なクラスを定義します。
 デフォルトのクラスが適切であれば、新しいクラスを定義する必要はありません。`audit_class(4)` のマニュアルページを参照してください。
2. `/etc/security/audit_event` 内でイベントとクラスのマッピングを設定します。
 デフォルトのマッピングがサイトのニーズに合っている場合は、この手順は不要です。`audit_event(4)` のマニュアルページを参照してください。
3. サイトに必要な監査の程度を決定します。
 サイトのセキュリティ上のニーズを考慮して、監査トレールの格納に必要なディスク領域を決定します。
 サイトのセキュリティを確保しながら記憶領域の要件を削減する方法と、監査記憶領域を設定する方法のガイドラインについては、23ページの「監査コストの制

御」、25ページの「効率的な監査」、27ページの「監査トレールについて」を参照してください。

4. どのマシンを監査サーバとして使用し、どのマシンを監査サーバのクライアントとして使用するかを決定します。
5. 監査ファイルシステムの名前と位置を決定します。
6. どのマシンが監査サーバ上のどの監査ファイルシステムを使用するかの計画を作成します。

記憶領域の計画を作成したら、監査の対象となるユーザと監査内容を決定します。

1. システム単位で監査したい監査クラスと、監査クラスの選択に使用するフラグを決定します。
2. 一部のユーザを他のユーザより詳しく監査するかどうかを決定し、ユーザの監査特性の変更に使用するフラグを決定します。
詳細は、15ページの「プロセスの監査特性」を参照してください。

3. 最小空き領域 (minfree) を決定します。これは弱い制限とも呼ばれ、警告が送られる前に監査ファイルシステム上に残っていなければならない空き領域です。使用可能な容量が minfree のパーセンテージを下回ると、監査デーモンは次の適切なファイルシステムに切り替えて、soft 制限を超えたことを示す通知を送ります (適切な監査ファイルシステムについては、18ページの「監査デーモンが使用できるディレクトリ」を参照してください)。

デフォルトでは、各マシン上で一定の程度の監査が構成されます。デフォルトの audit_control ファイルには、表 2-6 の各行が入っています。各行によって、監査ディレクトリが /var/audit として設定され、システムワイドな監査フラグ (lo) が 1 つ設定され、minfree のしきい値が 20 パーセントに設定され、非帰因フラグが 1 つ設定されます。

表 2-6 audit_control ファイル内のエントリ

```
dir:/var/audit
flags:lo
minfree:20
naflags:ad
```

表 2-6 audit_controlファイル内のエントリ 続く

4. /etc/security/audit_control ファイルを編集します。
 - a. このマシン上でどの監査ファイルシステムを監査トレールの格納に使用するかを指定します。

監査ディレクトリごとに dir: エントリをカレントマシンで使用できるようにします。分散システムの監査ディレクトリ方式を設定する方法については、27ページの「監査トレールについて」を参照してください。
 - b. flags: フィールド内で、すべてのユーザのプロセスに適用されるシステムワイドな監査フラグを指定します。

flags: フィールドで指定したシステムワイドな監査フラグは、すべてのユーザのプロセスに適用されるので、各マシン上で同じフラグを設定する必要があります。
 - c. 必要であれば、minfree のパーセンテージを変更して監査のしきい値を拡大または縮小します。
 - d. 特定のユーザに帰因しないイベントに適用される naflags: を指定します。
5. 変更が必要であれば、auditconfig を使用して監査方針を変更します。

auditconfig(1M) のマニュアルページ、または 38ページの「auditconfig コマンド」を参照してください。方針を指定する変数は動的なカーネル変数なので、その値はシステムの終了時に保存されません。したがって、適切な起動スクリプトを使用して目的の方針を設定する必要があります。
6. cnt 方針を設定するか、または監査管理アカウントを設定します。

監査トレールがあふれた場合に備えて、システムを引き続き機能させられるように cnt 方針を使用可能にするか、または監査されなくても機能できるようにアカウントを1つ使用可能にしなければなりません。このアカウントを設定する手順は次のとおりです。

 - a. /etc/passwd ファイルに次のエントリを追加します。

```
audit::0:1:::/sbin/sh
```

注 - このエントリは、root が所有するプロセスを正常に機能させるために、root ユーザのエントリの下に追加しなければなりません。

- b. 対応するエントリを /etc/shadow ファイルに追加するには、次のように入力します。

```
# pwconv
pwconv: WARNING user audit has no password
```

監査アカウントのパスワードは手順 d. で設定します。

- c. /etc/security/audit_user ファイルに次のエントリを追加して、このアカウントの監査をオフにします。

```
audit:no:all
```

- d. passwd を使用して新しいアカウントのパスワードを設定します。

```
# passwd audit
```

このアカウントを通じて実行される処置は監査の対象外なので注意してください。システムの完全性を保護するには、簡単には推測できないパスワードを使用してください。この例では、アカウント名として audit を使用しています。アカウントを設定する場合は、サイトに適した名前を選択してください。

監査トレールのオーバーフローを防ぐ

すべての監査ファイルシステムがいっぱいになると、audit_warn はすべての監査ファイルシステム上で強い制限値を超えたことを示すメッセージをコンソールに送り、別名にもメールを送ります。デフォルトで、監査デーモンはループ内に残り、ある程度の領域が解放されるまで休眠して領域の有無をチェックします。監査対象の処置はすべて中断されます。

サイトのセキュリティ要件の関係で、監査トレールのオーバーフローによってシステムの動作が中断されるよりは、一部の監査データが失われる方がよいという場合があります。その場合は、自動検出機能を構築するか、ファイルを `audit_warn` スクリプトに移動するか、または `auditconfig` を設定してレコードを削除させることができます。

▼ 監査トレールのオーバーフローを防ぐ方法

セキュリティ方針の関係ですべての監査データを保存する必要がある場合は、次の手順にしたがいます。

1. 定期的に監査ファイルを保存し、保存した監査ファイルを監査ファイルシステムから削除するようなスケジュールを設定します。
2. バックアップをテープに作成するか、保存ファイルシステムに移動して、監査ファイルを手作業で保存します。
3. 監査レコードの解釈に必要な、内容に対応する情報を、監査トレールといっしょに格納します。
4. どんな監査ファイルを移動したかを示すレコードをオフラインで保管します。
5. 保存したテープを適切な方法で保管します。
6. サマリファイルを作成して、格納する監査データのボリュームを削減します。
`auditreduce` のオプションを使用すると、監査トレールからサマリファイルを抽出できるため、サマリファイルには指定したタイプの監査イベントのみが入っています。たとえば、すべてのログインとログアウトの監査レコードのみが入ったサマリファイルがあります。第3章を参照してください。

auditconfig コマンド

`autoconfig` コマンドは、監査構成パラメータを設定するためのコマンド行インタフェースを提供します。`auditconfig(1M)` のマニュアルページを参照してください。`auditconfig` コマンドには次のようなオプションを使用できます。

-chkconf

カーネル監査イベントとクラスのマッピングの構成をチェックし、矛盾があれば報告します。

-conf

カーネルイベントとクラスのマッピングが、audit_event ファイル内の現在のマッピングと一致するように実行時に再構成します。

-getcond

マシンの監査条件を取得します。表 2-7 に考えられる応答を示します。

表 2-7 考えられる監査条件

| 応答 | 意味 |
|----------|----------------------|
| auditing | 監査が使用可能でオンになっている。 |
| no audit | 監査は使用可能だがオフになっている。 |
| disabled | 監査モジュールは使用可能になっていない。 |

-setcond *condition*

マシンの監査条件を auditing または noaudit で設定します。

-getclass *event_number*

指定するイベントがマップされている事前選択クラスを取得します。

-setclass *event_number audit_flags*

指定するイベントがマップされる事前選択クラスを設定します。

-lsevent

現在構成されている (実行時) カーネルとユーザ監査イベント情報を表示します。

`-getpinfo pid`

指定するプロセスの監査 ID、事前選択マスク、端末 ID、監査セッション ID を取得します。

`-setpmask pid flags`

指定するプロセスの事前選択マスクを設定します。

`-setsmask asid flags`

指定する監査セッション ID を持つすべてのプロセスの事前選択マスクを設定します。

`-setumask auid flags`

指定するユーザ監査 ID を持つすべてのプロセスの事前選択マスクを設定します。

`-lspolicy`

監査方針のリストと、それぞれの短い説明を表示します。

`-getpolicy`

現在の監査方針フラグを取得します。

`-setpolicy policy_flag[,policy_flag]`

監査方針フラグを指定する方針に設定します。次の監査方針の設定を参照してください。

監査方針の設定

`-setpolicy` フラグを指定して `auditconfig` を使用すると、デフォルトの Solaris-BSM 監査方針を変更できます。`-lspolicy` 引数を指定して `auditconfig` コマンドを使用すると、変更できる監査方針が表示されます。次のような方針フラグがあります。

arge

execv に関する環境変数を記録します (exec (2) のマニュアルページを参照)。デフォルトではこの情報を記録しません。

argv

execv のコマンド行引数を記録します。デフォルトではこれらの情報を記録しません。

cnt

待ち行列がいっぱいになっても、監査対象の動作を中断せず、単に削除された監査レコード数をカウントします。デフォルトでは中断します。

group

監査レコードに補助グループトークンを含ませます。デフォルトでは、グループトークンは含まれません。

path

監査レコードに二次 path トークンを追加します。一般に、これらの二次パスは、動的にリンクされた共有ライブラリまたはシェルスクリプトのコマンドインタープリタのパス名です。デフォルトでは、これらは含まれません。

trail

すべてのレコードに trailer トークンが含まれます。デフォルトでは、trailer トークンは記録されません。

seq

すべての監査レコードにシーケンス番号が含まれます。デフォルトでは、シーケンス番号は含まれません (シーケンス番号を使用すると、クラッシュダンプを分析して監査レコードが失われたかどうかを調べることができます)。

▼ どのイベントがどの監査クラスに属するかを変更する方法

次の手順で、デフォルトのイベントとクラスとのマッピングを変更します。

1. /etc/security/audit_event ファイルを編集して、目的の各イベントのクラスマッピングを変更します。
2. システムをリブートするか、または auditconfig -conf を実行して、実行時カーネルイベントとクラスとのマッピングを変更します。

クラス定義の変更

ファイル /etc/security/audit_class には、クラス定義が格納されます。サイト固有の定義を追加して、デフォルトの定義を変更できます。このファイル内の各エントリの書式は次のとおりです。

mask:name:description

各クラスはマスク内の 1 ビットとして表されます。これは符号なしの整数で、32 種類の使用可能なクラスと、2 つのメタクラス all と no を示します。all は、使用可能なすべてのクラスを連結したものです。no は無効なクラスです。このクラスにマップされたイベントは監査されません。no クラスにのみマップされたイベントは、all クラスがオンになっていても監査されません。次は、audit_class ファイルの例です。

```
0x00000000:no:invalid class
0x00000001:fr:file read
0x00000002:fw:file write
0x00000004:fa:file attribute access
0x00000008:fm:file attribute modify
0x00000010:fc:file create
0x00000020:fd:file delete
0x00000040:cl:file close
0xffffffff:all:all classes
```

システムカーネル内で no クラスがオンになっていると、監査トレールは監査イベント AUE_NULL のレコードでいっぱいになってしまいます。

監査トレールの分析

この章で説明するツールを使用すると、監査ファイルを管理して報告するシェルスクリプトを開発し、それを定期的に行うことができます。一般的に監査管理作業には、ファイルを圧縮したり、複数の監査ファイルを組み合わせることで1つにまとめたり、ファイルを分散システム内でディスク上のさまざまな位置に移動したり、古いファイルをテープに保存することなどが含まれます。また、スクリプトを使用して記憶領域の使用状況を監視できますが、その一部は監査デーモンによって自動的に実行されます。

もう1つの監査作業は、すべての監査ファイルを論理的に組み合わせた監査トレールを検査することです。監査ツールを使用すると、監査データファイル内の特定の情報を対話形式で照会できます。

- 43ページの「監査機能」
- 44ページの「監査レコードのマージ、選択、表示、変換に使用するツール」
- 45ページの「監査レコードの形式」
- 55ページの「auditreduce コマンドの使用法」
- 58ページの「praudit の使用法」

監査機能

Solaris BSM には、監査レコードを理解できるように次の機能が組み込まれています。

- ユーザのプロセスに割り当てられた監査 ID は、ユーザ ID が変更されても変わりません。

- 各セッションは監査セッション ID を持ちます。
- 監査レコードに完全パス名が記録されます。

各監査レコードには、イベントを生成したユーザを識別する監査 ID が入っているため、完全パス名も記録されるので、監査トレール全体を見直さなくても、個々の監査レコードを調べて有用な情報を得ることができます。

監査ユーザ ID

Solaris BSM のプロセスには、標準 Solaris リリースのプロセスには関連付けられない、別のユーザ識別属性、つまり監査 ID が付いています。プロセスはログイン時に監査 ID を取得し、この監査 ID はすべての子プロセスに継承されます。

監査セッション ID

Solaris BSM のプロセスには、ログイン時に監査セッション ID が割り当てられます。この ID は、すべての子プロセスに継承されます。

それ自身で意味のわかる監査レコード

Solaris BSM の監査レコードにはイベントに関連するすべての情報が入っているため、他の監査レコードを参照しなくても発生したイベントを理解できます。たとえば、ファイルイベントを記述する監査レコードには、そのファイルに関してルートディレクトリから始まる完全パス名と、オープンまたはクローズした日時を表すスタンプが入っています。

監査レコードのマージ、選択、表示、変換に使用するツール

Solaris BSM には、監査レコードのマージ、選択、表示、変換に使用できるように 2 つのツールが組み込まれています。これらのツールを直接使用するか、サードパーティのアプリケーションプログラムと併用できます。

- `auditreduce` コマンドを使用すると、検証したいレコードのセットを選択できます。たとえば、過去 24 時間分のすべてのレコードを選択して日次レポートを

生成したり、特定のユーザによって生成されたすべてのレコードを選択して、そのユーザによる動作を検証できます。さらに、特定のイベントタイプによるすべてのレコードを選択して、そのタイプの発生頻度を調べることもできます。

- `praudit` コマンドを使用すると、監査レコードを対話形式で表示し、基本的なレポートを作成できます。`praudit` は、通常ユーザが読めない形式のレコードを、ユーザが読めるいくつかの形式のいずれかで表示します。`praudit` からの出力を (`sed` や `awk` などを使用して) 後処理するか、または 2 進の監査レコードを変換して処理するプログラムを作成して、さらに複雑な内容を表示したりレポートに作成したりできます。

この後の各節は、監査レコードの形式で、`praudit`、`auditreduce` コマンドの詳細、ツールを使うためのヒントと手順について説明します。

監査レコードの形式

Solaris BSM の監査レコードは、一連の監査トークンからなっており、各トークンでシステムの属性が記述されます。

各監査トークンについての詳細は 付録 A を参照してください。この付録には、Solaris BSM の監査機能によって生成される全監査レコードのリストも掲載されます。定義は短い説明の順にソートされていて、相互参照一覧ではイベント名とその説明箇所が示されています。

2 進形式

監査レコードは 2 進形式で格納され処理されますが、さまざまなマシン間での互換性を保つために、データのバイトオーダーとサイズはあらかじめ決められています。

監査イベントのタイプ

システム内の各監査対象イベントによって、特定のタイプの監査レコードが生成されます。各イベントの監査レコードには、そのイベントを記述する特定のトークンが入っています。監査レコードには、イベントが属する監査イベントクラスは記述されません。そのマッピングは外部テーブル、`/etc/security/audit_event` ファイルによって指定されます。

監査トークンのタイプ

各トークンは1バイトのトークンタイプから始まり、タイプ別に決められた順序で1つまたは複数のデータ要素が続いています。監査レコードの種類は、そのレコード内のイベントタイプと各種トークンセットによって区別されます。text トークンのようにデータ要素が1つしか入っていないトークンと、process トークンのように複数の要素(監査ユーザ ID、実ユーザ ID、実効ユーザ ID など)が入っているトークンがあります。

監査トークンの順序

各監査レコードは、header トークンで始まって、trailer トークン(省略可能)で終わります。ヘッダとトレーラの間にある1つまたは複数のトークンでイベントが記述されます。ユーザレベルのイベントとカーネルイベントの場合は、トークンでそのイベントを実行したプロセス、実行対象となったオブジェクト、所有者やモードなどのオブジェクトのトークンが記述されます。

一般に、それぞれのユーザレベルイベントとカーネルイベントには、少なくとも次のトークンが付いています。

- header
- subject
- return

trailer トークンは多くのイベントに含まれていますが、それは省略可能です。

ユーザが読める監査レコード書式

この節では、各監査レコードの書式を `praudit` コマンドで生成される出力どおりに示し、各監査トークンについて簡単に説明します。各トークン内のフィールドについての詳細は付録 A を参照してください。

次のトークンの例は、デフォルトで `praudit` によって生成される書式を示しています。また、各例は `raw(-r)` オプションと短縮 `(-s)` オプションを指定することが前提となっています。`praudit` によって監査トークンが表示されるときは、まずトークンタイプ、次にそのトークンからのデータが表示されます。ただし、フィールド(パス名など)にカンマが入っている場合は、それとフィールド区切りのカンマとを区別できません。別のフィールド区切り記号を使用しないと、出力にカンマが含ま

れることとなります。デフォルトでは、トークンタイプは header のように名前として表示されるか、または 10 進数として -r 形式で表示されます。

各トークンを次の順序で説明します。

- 47ページの「header トークン」
- 48ページの「trailer トークン」
- 48ページの「arbitrary トークン」
- 49ページの「arg トークン」
- 49ページの「attr トークン」
- 50ページの「exit トークン」
- 50ページの「file トークン」
- 50ページの「groups トークン」
- 50ページの「in_addr トークン」
- 51ページの「ip トークン」
- 51ページの「ipc トークン」
- 51ページの「ipc_perm トークン」
- 52ページの「iport トークン」
- 52ページの「opaque トークン」
- 52ページの「path トークン」
- 53ページの「process トークン」
- 53ページの「return トークン」
- 53ページの「seq トークン」
- 54ページの「socket トークン」
- 54ページの「subject トークン」
- 55ページの「text トークン」

header トークン

すべての監査レコードは header トークンで始まります。header トークンは、すべての監査レコードに共通の情報を示します。次のフィールドが入っています。

- トークン ID

- header トークンと trailer トークンを含めたバイト単位のレコード長
- 監査レコード構造体のバージョン番号
- 監査イベントのタイプを識別するイベント ID
- イベントタイプに関する記述情報が付いたイベント ID 修飾子
- レコードの作成日時

header トークンが `praudit` によってデフォルト形式で表示されるときは、次の `ioctl` からの例のようになります。

```
header,240,1,ioctl(2),es,Tue Sept 1 16:11:44 1992, + 270000 msec
```

`praudit -s` を使用すると、イベント記述 (`ioctl(2)`) は、次のようにイベント名 (`AUE_IOCTL`) に置き換えられます。

```
header,240,1,AUE_IOCTL,es,Tue Sept 1 16:11:44 1992, + 270000 msec
```

`praudit -r` を使用すると、すべてのフィールドが数値として表示されます (10 進数、8 進数、または 16 進数)。この場合、158 はこのイベントのイベント番号です。

```
20,240,1,158,0003,699754304, + 270000 msec
```

`praudit` では時刻がミリ秒単位で表示されるので注意してください。

trailer トークン

このトークンは監査レコードの終りを示し、監査トレールを逆方向から検索できるようにします。次のフィールドがあります。

- トークン ID
- レコードの終わりを示すパッド番号 (表示されない)
- header トークンと trailer トークンを含めた監査レコードの合計文字数

`praudit` により、trailer トークンは次のように表示されます。

```
trailer,136
```

arbitrary トークン

このトークンは、監査トレールのデータをカプセル化します。項目の配列には多くの項目が含まれることがあります。次のフィールドがあります。

- トークン ID
- 10 進などの推奨形式

- int など、カプセル化されたデータのサイズ
- データ配列の項目数
- 項目の配列

praudit により、arbitrary トークンは次のように表示されます。

```
arbitrary,decimal,int,1
42
```

arg トークン

このトークンには、システムコールの引数情報が入っています。監査レコード内では、32 ビット整数によるシステムコール引数を使用できます。次のフィールドがあります。

- トークン ID
- 関連するシステムコール引数の引数 ID
- 引数の値
- 省略可能な記述テキスト文字列の長さ (表示されない)
- 省略可能なテキスト文字列

praudit により、arg トークンは次のように表示されます。

```
argument,1,0x00000000,addr
```

attr トークン

一般に、attr トークンはパスの検索中に生成され、path トークンが添付されていますが、パス検索エラーのイベントには含まれません。次のフィールドがあります。

- トークン ID
- ファイルのアクセスモードとタイプ
- 所有者のユーザ ID
- 所有者のグループ ID
- ファイルシステム ID
- i ノード ID
- ファイルが示すデバイス ID

praudit により、attr トークンは次のように表示されます。

```
attribute,100555,root,staff,1805,13871,-4288
```

exit トークン

exitトークンには、プログラムの終了状態が記録されます。次のフィールドがあります。

- トークン ID
- exit() システムコールに渡されるプログラムの終了状態
- 終了状態を記述するか、システムエラー番号を示す戻り値

praudit により、exit トークンは次のように表示されます。

```
exit,Error 0,0
```

file トークン

このトークンは監査デーモンによって生成され、古いファイルが有効でなくなると、新しい監査トレールファイルのはじめと古いファイルの終りを示します。このトークンが入っている監査レコードは、連続する監査ファイルをまとめてリンクし、1つの監査トレールにまとめます。次のフィールドがあります。

- トークン ID
- ファイルのオープンまたはクローズ日時を示すスタンプ
- ファイル名のバイト数 (表示されない)
- ファイル名

praudit により、file トークンは次のように表示されます。

```
file,Tue Sep 1 13:32:42 1992, + 79249 msec,  
/baudit/localhost/files/19920901202558.19920901203241.quisp
```

groups トークン

groupsトークンは、プロセスの資格からグループエントリを記録します。次のフィールドがあります。

- トークン ID
- サイズが NGROUPS_MAX (16) のグループエントリの配列

praudit により、groups トークンは次のように表示されます。

```
group,staff,wheel,daemon,kmem,bin,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
```

in_addr トークン

in_addrトークンは、マシンのインターネットプロトコルアドレスを示します。次のフィールドがあります。

- トークン ID
- インターネットアドレス

praudit により、in_addr トークンは次のように表示されます。

```
ip addr,129.150.113.7
```

ip トークン

ip トークンには、インターネットプロトコルヘッダのコピーが入っています。次のフィールドがあります。

- トークン ID
- IP ヘッダの 20 バイトのコピー

praudit により、ip トークンは次のように表示されます。

```
ip address,0.0.0.0
```

ipc トークン

このトークンには、特定の IPC オブジェクトを識別するために呼び出し元に使用される System V IPC メッセージ/セマフォ/共有メモリハンドルが入っています。次のフィールドがあります。

- トークン ID
- IPC オブジェクトタイプ識別子
- IPC オブジェクトハンドル

praudit により ipc トークンは次のように表示されます。

```
IPC,msg,3
```

ipc_perm トークン

ipc_perm トークンには、System V IPC アクセス情報のコピーが入っています。共有メモリ、セマフォ、メッセージ IPC の監査レコードには、このトークンが追加されます。次のフィールドがあります。

- トークン ID
- IPC 所有者のユーザ ID
- IPC 所有者のグループ ID
- IPC 作成者のユーザ ID
- IPC 作成者のグループ ID

- IPC アクセスモード
- IPC シーケンス番号
- IPC キー値

praudit により ipc_perm トークンは次のように表示されます。

```
IPC perm,root,wheel,root,wheel,0,0,0x00000000
```

ipport トークン

このトークンには、TCP (または UDP) アドレスが入っています。次のフィールドがあります。

- トークン ID
- TCP/UDP アドレス

praudit により ipport トークンは次のように表示されます。

```
ip port,0xf6d6
```

opaque トークン

opaque トークンには、書式化されていないデータが一連のバイトとして入っています。次のフィールドがあります。

- トークン ID
- データ配列のバイト数
- バイトデータ配列

praudit により opaque トークンは次のように表示されます。

```
opaque,12,0x4f5041515545204441544100
```

path トークン

path トークンには、オブジェクトに関するアクセスパス情報が入っています。次のフィールドがあります。

- トークン ID
- パス長のバイト数 (表示されない)
- 絶対パス

praudit により path トークンは次のように表示されます。

```
path,/an/anchored/path/name/to/test/auditwrite/AW_PATH
```


process トークン

process トークンには、プロセスを記述する情報が入っています。次のフィールドがあります。

- トークン ID
- ユーザの監査 ID
- 実効ユーザ ID
- 実効グループ ID
- 実ユーザ ID
- 実グループ ID
- プロセス ID
- セッション ID
- 次の ID からなる端末 ID
 - デバイス ID
 - マシン ID

praudit により、process トークンは次のように表示されます。

```
process,root,root,wheel,root,wheel,0,0,0,0.0.0.0
```

return トークン

return トークンは、システムコールの戻り状態とプロセスの戻り値を示します。このトークンは、常にシステムコールに関してカーネルで生成される監査レコードの一部として返されます。次のフィールドがあります。

- トークン ID
- システムコールのエラー状態
- システムコールの戻り値

praudit により、return トークンは次のように表示されます。

```
return,success,0
```

seq トークン

このトークンは省略可能で、デバッグに使用される昇順のシーケンス番号が入っています。seq ポリシーがアクティブになっているときは、このトークンが各監査レコードに追加されます。次のフィールドがあります。

- トークン ID
- 32 ビットの符号なし long 型のシーケンス番号

praudit により、seq トークンは次のように表示されます。

```
sequence,1292
```

socket トークン

socket トークンはインターネットソケットを記述します。次のフィールドがあります。

- トークン ID
- ソケットタイプフィールド (TCP/UDP/UNIX)
- ローカルポートアドレス
- ローカルのインターネットアドレス
- リモートポートアドレス
- リモートのインターネットアドレス

praudit により、socket トークンは次のように表示されます。

```
socket,0x0000,0x0000,0.0.0.0,0x0000,0.0.0.0
```

subject トークン

このトークンはサブジェクト (プロセス) を記述します。次のフィールドがあります。

- トークン ID
- ユーザの監査 ID
- 実効ユーザ ID
- 実効グループ ID
- 実ユーザ ID
- 実グループ ID
- プロセス ID
- セッション ID
- 次の ID からなる端末 ID
 - デバイス ID

- マシン ID

praudit により、subject トークンは次のように表示されます。

```
subject,cjc,cjc,staff,cjc,staff,424,223,0 0 quisp
```

text トークン

text トークンにはテキスト文字列が入っています。次のフィールドがあります。

- トークン ID
- テキスト文字列の長さ (表示されない)
- テキスト文字列

praudit により、text トークンは次のように表示されます。

```
text,aw_test_token
```

auditreduce コマンドの使用法

auditreduce コマンドを実行すると、1 つまたは複数の入力監査ファイルから監査レコードがマージされます。通常は、分散システム全体のすべての監査トレールファイルがマウントされているマシンから、このコマンドを入力することになります。

オプションを指定せずに auditreduce を実行すると、監査される全体 (監査ディレクトリ /etc/security/audit 内のすべてのサブディレクトリ内のすべての監査ファイル) がマージされ、そのマージされたファイルが標準出力に送られます。

praudit コマンドを実行すると、各レコードはユーザが読める形式に変換されます。詳細は 58 ページの「praudit の使用方法」を参照してください。

auditreduce コマンドのオプションには、次のような機能があります。

- 特定の監査フラグでのみ生成された監査レコードが入っている出力が表示される。
- 特定の 1 人のユーザによって生成された監査レコードが表示される。
- 特定の日付に生成された監査レコードが収集される。

分散システムで auditreduce を役立てる方法

Solaris BSM を実行する複数のマシンが分散システムの一部として管理される場合には、各マシンが監査対象のイベントを実行し、監査レコードをマシン固有の専用監査ファイルに書き込みます。このため、ソフトウェアが単純化され、マシンの障害が発生した場合の信頼性が高まります。

auditreduce コマンドによって、監査トレール全体の管理作業を効率化できます。auditreduce (または、より高いインタフェースを提供するために独自に作成したシェルスクリプト) を使用すると、レコードの生成方法や格納場所に関係なく、システム内のすべてのファイルの論理上の組み合わせを 1 つの監査トレールとして読むことができます。

auditreduce プログラムは、監査デーモンによって生成された監査トレールを処理します。1 つまたは複数の監査ファイルからレコードが選択され、マージされて、1 つの時系列順のファイルが生成されます。auditreduce のマージ機能と選択機能は論理的に他に依存しません。auditreduce はレコードが読み取られると、入力ファイルがマージされてディスクに書き込まれる前に、そこからメッセージを選択します。auditreduce (1M) のマニュアルページを参照してください。

auditreduce の使用方法

この節では、データを分析して管理するための auditreduce の一般的な使用方法について説明します。

監査ログ全体を表示する方法

監査トレール全体を一度に表示するには、auditreduce の出力を praudit にパイプします。

```
# auditreduce | praudit
```

監査ログ全体を印刷する方法

lp にパイプすると、出力はプリンタに送られます。

```
# auditreduce | praudit | lp
```

選択したデータに関するユーザの動作を表示する方法

次の例で、システム管理者は `lo` メッセージクラスを要求して、ユーザ `fred` が 1990 年 4 月 13 日にログインしてログアウトした時刻を調べます。短い日付の書式は `yyymmdd` です (長い形式については、`auditreduce(1M)` のマニュアルページを参照)。

```
# auditreduce -d 900413 -u fred -c lo | praudit
```

ログイン/ログアウトメッセージを 1 つのファイルにコピーする方法

次の例では、特定の日付に関するログイン/ログアウトメッセージが 1 つのファイルに集計されます。ターゲットファイルは、通常の監査ルート以外のディレクトリに書き込まれます。

```
# auditreduce -c lo -d 870413 -O /usr/audit_summary/logins
```

`-O` オプションを使用すると、開始時刻と終了時刻を示す 14 文字のタイムスタンプと接尾辞 `logins` が付いた監査ファイルが作成されます。

```
/usr/audit_summary/19870413000000.19870413235959.logins
```

not_terminated 監査ファイルを整理する方法

監査ファイルが開いているうちに監査デーモンが停止したり、サーバがアクセスできなくなってマシンが新しいサーバに強制的に切り替えたりすると、監査ファイルが監査レコードに使用されなくなっても、そのファイルが残り、ファイル名に含まれる終了時刻に文字列 `not_terminated` が付いたままになることがあります。この種のファイルが検出された場合は、ファイルが使用されていないことを手作業で検査し、正しいオプションを使用してファイル名を指定して整理するとよいでしょう。

```
# auditreduce -O machine 19870413120429.not_terminated.machine
```

このコマンドを実行すると、正しい名前 (両方のタイムスタンプ) と正しい接尾辞 (明示的に指定された `machine`) が付いた新しい監査ファイルが作成され、すべてのメッセージがそのファイルにコピーされます。

その他の有用な auditreduce オプション

`auditreduce` には他にも多数のオプションがあり、マニュアルページに掲載されています。大文字のオプションを使用すると `files` に対する処理またはパラメータ

が選択され、小文字のオプションを使用すると records に対するパラメータが選択されるので注意してください。この項では、2つの便利なオプションの使用方法について説明します。

date-time オプション `-b` と `-a` を使用すると、特定の日時よりも前または後のレコードを指定できます。1日は `yyyymmdd00:00:00` から始まって `yyyymmdd23:59:59` に終わります。日付に関するパラメータは年、月、日、時、分、秒の6つです。年のはじめの2桁(19)は指定する必要はありません。

`-a` を指定しなければ、`auditreduce` はデフォルトの1970年1月1日00:00:00となります。`-b` を指定しなければ、`auditreduce` はデフォルトの現在の日時(GMT)となります。57ページの「ログイン/ログアウトメッセージを1つのファイルにコピーする方法」を参照してください。

次のように日付を指定して `auditreduce -a` コマンドを実行すると、1991年7月15日の午後12時以降に作成されたすべての監査レコードが `praudit` に出力されます。

```
# auditreduce -a 91071500:00:00 | praudit
```

上記と同じ日付を指定して `auditreduce -b` コマンドを実行すると、1991年7月15日の午後12時以前に作成されたすべての監査レコードが `praudit` に出力されます。

```
# auditreduce -b 91071500:00:00 | praudit
```

`auditreduce` のメッセージタイプ選択 (`-m` オプション) では、数値メッセージ識別子または `AUE_xxxxx` コマンドを指定できます。間違った書式を指定すると `auditreduce` に拒否されますが、正しい書式は表示されません。

praudit の使用方法

`praudit` コマンドは、標準入力から監査レコードを読み取り、ユーザが読める書式で標準出力に表示します。一般に、入力は `auditreduce` からパイプされるか、または1つの監査ファイルです。また、`cat` を使用して入力を生成して複数のファイルを連結したり、現在の監査ファイルに `tail` を使用することもできます。

`praudit` を使用すると、デフォルト形式、短い形式 (`-s` オプション)、`raw` 形式 (`-r` オプション) という3つの形式で出力を生成できます。デフォルトでは、出力は

1 行に 1 つずつトークンが入る形式で生成されます。-l オプションでは、各行にレコード全体が入るように要求します。-d オプションでは、トークンフィールド間に使用される区切り記号を変更し、-l も指定すればトークン間に使用される区切り記号も変更されます。

-s 形式では、タイプはイベントの監査イベントテーブル名 (AUE_IOCTL など) で -r 形式ではイベント番号 (この場合は 158) です。これは、-s とデフォルト形式の唯一の違いです。-r 形式では、すべての数値 (ユーザ ID、グループ ID など) は数値で表示されます (10 進形式、ただしインターネットアドレスの場合は 16 進形式、モードの場合は 8 進形式)。次の例は、header トークンに関する praudit からの出力を示しています。

```
header,240,1,ioc1(2),es,Tue Sept 1 16:11:44 1992, + 270000 msec
```

次の例は、同じ header トークンに関する praudit -r からの出力を示しています。

```
20,240,1,158,0003,699754304, + 270000 msec
```

auditreduce では実行できない選択を行う場合など、出力をテキスト行として処理する方が便利な場合があります。praudit の出力は単純なシェルスクリプトで処理できます。次の例は praudit_grep というスクリプトです。

```
#!/bin/sh
praudit | sed -e '1,2d' -e '$s/^file.*$/' -e 's/^header/^aheader/' \
| tr '\012\001' '\002\012' \
| grep "$1" \
| tr '\002' '\012'
```

このスクリプト例は、header トークンに接頭辞として Ctrl-A を付けて示します (^a は ^ と a という 2 つの文字ではなく Ctrl-A を示すので注意してください。接頭辞は、header トークンをテキストとして表示される文字列 header と区別するために必要です)。このスクリプトは、改行を Ctrl-A として残したままでレコードのすべてのトークンを組み合わせて 1 行にまとめ、grep を実行し、元の改行を復元します。

praudit のデフォルトの出力形式では、各レコードを header トークンで始まって trailer トークンで終わる一連のトークン (1 行に 1 つずつ) として常に明確に識別できるので注意してください。したがって、各レコードを簡単に識別し、awk などを使用して処理できます。

デバイスの割り当て

C2 レベル以上のコンピュータシステムに関する Trusted Computer System Evaluation Criteria (TCSEC) のオブジェクト再使用要件は、デバイス割り当て機構によって達成されます。この章では、デバイスを管理するうえで必要な情報について説明します。

割り当て可能にするべきデバイスがあるかどうか、そしてあるとすれば、デフォルトの割り当てがサイトのセキュリティポリシーに適していない場合は、どのデバイスを割り当ててのかを決定する必要があります。

- 62ページの「デバイスの使用に伴うリスク」
- 62ページの「デバイス割り当て機構の構成要素」
- 63ページの「デバイス割り当てユーティリティの使用方法」
- 64ページの「割り当てエラー状態」
- 65ページの「device_maps ファイル」
- 66ページの「device_allocate ファイル」
- 69ページの「デバイスクリーンسكريプト」
- 71ページの「ロックファイルの設定」
- 74ページの「デバイスの管理と追加」
- 75ページの「デバイス割り当ての使用方法」

デバイスの使用に伴うリスク

各種 I/O デバイスの使用に伴うセキュリティ上のリスクの一例として、一般にカートリッジデバイスがどのように使用されるかを考えてみてください。通常は複数のユーザが1つのテープドライブを共有しますが、それは各ユーザのマシンが配置されている場所から離れたオフィスや研究所に配置されていることがあります。これは、ユーザがテープをテープドライブにロードしてから、マシンに戻ってテープとの間でデータを読み書きするコマンドを起動するまでに、ある程度の時間が経過することを意味します。また、ユーザが戻ってテープをドライブから取り出すまでもにも時間が経過します。テープデバイスにはユーザ全員がアクセスできるのが普通なので、テープのそばにユーザがいない間に、権限を持たないユーザがテープ上のデータにアクセスしたり、上書きしたりする可能性があります。

デバイス割り当て機構によって、特定のデバイスを一度に1人のユーザに割り当てることができます。これにより、デバイスが特定のユーザの名前に割り当てられている間は、そのユーザしかアクセスできなくなります。

デバイス割り当て機構によって、テープデバイスに関して次のことが保証され、それに関連するセキュリティサービスが他の割り当て可能デバイスに提供されます。

- デバイスへの同時アクセスが防止される。
- あるユーザがテープドライブからテープを取り出すまで、そのユーザは別のユーザが書き込んだばかりのテープは読み取れなくなる。
- ユーザがデバイスを使用し終わった後は、別のユーザはそのデバイスやドライブの内部記憶領域から情報を収集できなくなる。

デバイス割り当て機構の構成要素

デバイス割り当てを管理するには、割り当て機構の次の構成要素を理解しておかなければなりません。

- `allocate`、`deallocate`、`dminfo`、`list_devices` コマンド
- `/etc/security/device_allocate` ファイル (`device_allocate(4)` のマニュアルページを参照)
- `/etc/security/device_maps` ファイル (`device_maps(4)` のマニュアルページを参照)

- /etc/security/dev 内に割り当て可能デバイスごとに存在しなければならないロックファイル
- 各割り当て可能デバイスに関連付けられたデバイス特殊ファイルの変更後の属性
- 各割り当て可能デバイスのデバイスクリーンスクリプト

ユーザが `allocate`、`deallocate`、`dminfo`、`list_devices` コマンドを起動する方法については、63ページの「デバイス割り当てユーティリティの使用法」を参照してください。すべてのオプションと他の記述の定義については、マニュアルページを参照してください。

`device_allocate` ファイル、`device_map` ファイル、ロックファイルは、各マシンに固有のものであります。テープドライブ、フロッピーディスクドライブ、プリンタは、すべて特定のマシンに接続されているので、構成ファイルが NIS データベースとして管理されることはありません。

デバイス割り当てユーティリティの使用法

この節では、`root` でなければ使用できない

`allocate`、`deallocate`、`list_devices` のオプションを使用して管理者が実行できる処理について説明します。各コマンドについての詳細は、それぞれのマニュアルページを参照してください。

`allocate -F device_special_filename`

指定するデバイスを再度割り当てます。通常は、このオプションを `-U` オプションと併用して、指定するデバイスを指定するユーザに再度割り当てます。`-U` オプションを指定しなければ、デバイスは `root` に割り当てられます。オプションを指定しなければ、デバイスは `root` に割り当てられます。

`allocate -U username`

デバイスは、現在のユーザではなく指定するユーザに割り当てられます。このオプションを使用すると、`root` になっている間は、指定するユーザの識別情報がなくても、そのデバイスを割り当てることができます。

`deallocate -F device_special_filename`

ユーザに割り当てられたデバイスは、プロセスが終了するとき、またはそのユーザがログアウトするときに、自動的に割り当て解除されます。ユーザがテープドライブの割り当てを解除し忘れたときには、`root` になっている間に `-F` オプションを使用して割り当てを解除させることができます。

`deallocate -I`

割り当て可能なすべてのデバイスの割り当てを強制的に解除させます。このオプションは、システムの初期化時にのみ使用してください。

`list_devices`

`list_devices` を実行して、`device_maps` ファイル内にリストされたデバイスに関連付けられたすべてのデバイス特殊ファイルのリストを取得します。

`list_devices -U username`

指定したユーザ名に関連付けられたユーザ ID に割り当て可能か割り当て済みのデバイスがリストされます。このため、`root` になっている間に、どのデバイスを別のユーザに割り当て可能か、または割り当て済みかをチェックできます。

割り当てエラー状態

割り当てエラー状態については、割り当て構成要素に関するマニュアルページを参照してください。割り当て可能デバイスは、デバイス特殊ファイルモードが `0100` になっているユーザ `bin` とグループ `bin` に所有されている場合に、割り当てエラー状態になります。ユーザが割り当てエラー状態になっているデバイスを割り当てたい場合は、`-F` オプションを指定して `deallocate` コマンドを使用して、そのデバイスの割り当てを強制的に解除するか、または `allocate -U` を使用してユーザに割り当ててから、表示されるエラーメッセージを調べる必要があります。デバイス関連の問題が解決されたら、`deallocate -F` または `allocate -F` コマンドをもう一度実行して、デバイスから割り当てエラー状態を解除しなければなりません。

device_maps ファイル

/etc/security/device_maps ファイルを調べると、各割り当て可能デバイスに関連付けられたデバイス名、デバイスタイプ、デバイス特殊ファイルを検索できます。device_maps(4) のマニュアルページを参照してください。デバイスマップは、デバイス割り当てを設定するときにシステム管理者によって作成されます。基礎となるファイルは、BSM が使用可能になると bsmconv によって作成されます。この初期マップファイルは、あくまでも出発点として使用する必要があります。このシステム管理者は、個々のサイトの device_maps を強化してカスタマイズすることができます。

このファイルでは、デバイスごとにデバイス特殊ファイルのマッピングが定義されます。多くの場合、このマッピングは単純ではありません。このファイルによって、各種のプログラムはどのデバイス特殊ファイルがどのデバイスにマップされているかを検出できます。たとえば、dminfo コマンドを使用すると、デバイスタイプとデバイス特殊ファイルを取得して、割り当て可能なデバイスを設定するときに指定できます。dminfo は device_maps ファイルを使用します。

各デバイスは、次の形式の 1 行のエントリで表されます。

device-name:device-type:device-list

このファイル内の各行の終わりに \ を付けて、エントリを次行に継続させることができます。また、コメントも挿入できます。# を付けると、それに続くすべてのテキストは、\ の直後にない次の改行までコメントになります。

どのフィールドにも先行ブランクと後続ブランクを使用できます。

device-name

st0、fd0、または audio などのデバイス名です。ここで指定するデバイス名は、/etc/security/dev ディレクトリ内で使用されるロックファイルの名前と対応しなければなりません。

device-type

汎用デバイスタイプ (st、fd、audio などのデバイスクラス名) です。device-type では、関連するデバイスが論理的にグループ化されます。

device-list

物理デバイスに関連付けられたデバイス特殊ファイルのリストです。*device-list* には、特定のデバイスにアクセスできるすべての特殊ファイルが入っていないければなりません。リストが不完全な場合は、悪意を持ったユーザでも個人情報を入手または変更できることとなります。また、次の例のように、バイナリ互換を保つために、`/devices` の下の実デバイスファイルや `/dev` 内のシンボリックリンクが *device-list* の有効なエントリであることに注意してください。

次の画面は、`device_maps` ファイル内の SCSI テープ `st0` とディスク `fd0` に関するエントリの例を示しています。

```
fd0:\
fd:\
/dev/fd0 /dev/fd0a /dev/fd0b /dev/fd0c /dev/rfd0 /dev/rfd0a\
/dev/rfd0b /dev/rfd0c:\
.
.
.
st0:\
st:\
/dev/rst0 /dev/rst8 /dev/rst16 /dev/rst24 /dev/nrst0 /dev/nrst8\
/dev/nrst16 /dev/nrst24:\
```

device_allocate ファイル

`device_allocate` ファイルを変更して、デバイスを割り当て可能から割り当て不可に変更したり、新しいデバイスを追加したりします。表 4-1 は、`device_allocate` ファイルのサンプルを示しています。

表 4-1 `device_allocate` ファイルのサンプル

| |
|---|
| <pre>st0;st;;;/etc/security/lib/st_clean fd0;fd;;;/etc/security/lib/fd_clean sr0;sr;;;/etc/security/lib/sr_clean audio;audio;;;*/etc/security/lib/audio_clean</pre> |
|---|

管理者は、基本セキュリティモジュールの初期構成中に、どのデバイスを割り当て可能にするかを定義します。表 4-1 のように、デフォルトのデバイスとその定義済み特性をそのまま使用するように決定できます。システムを稼働させた後の実行中

にマシンにデバイスを追加するときには、新しいデバイスを割り当て可能にするかどうかを決定しなければなりません。

管理者は、デバイスをインストールした後に、`device_allocate` ファイル内でそのデバイスのエントリを変更できます。割り当てたいでデバイスは、使用する前に各マシン上の `device_allocate` ファイル内で定義しなければなりません。現在、カートリッジテープドライブ、フロッピーディスクドライブ、CD-ROM デバイス、オーディオチップは割り当て可能と見なされ、デバイスクリーンスクリプトが用意されています。

注 - Xylogics テープドライブまたは Archive テープドライブを追加する場合は、SCSI デバイス用に用意されている `st_clean` スクリプトも使用できます。他には、モデム、端末、グラフィックスタブレットなどのデバイスを割り当て可能にすることができますが、この種のデバイスの場合は独自のデバイスクリーンスクリプトを作成する必要があります。また、その場合、スクリプトはそのタイプのデバイスのオブジェクト再使用要件を満たさなければなりません。

`device_allocate` ファイル内のエントリは、デバイスが割り当て可能であると特に記述されていない限り、そのデバイスが割り当て可能であることを意味しません。表 4-1 で、オーディオデバイスエントリの第 5 フィールドにアスタリスク (*)が入っているので注意してください。第 5 フィールド内のアスタリスクは、そのデバイスが割り当て可能でないことをシステムに指示します。つまり、システム管理者はユーザにデバイスを使用する前に割り当てたり、後で割り当てを解除するように要求する必要がありません。このフィールド内の他の文字列は、デバイスが割り当て可能であることを示します。

`device_allocate` ファイル内で、次の書式の 1 行のエントリで各デバイスを表します。

```
device-name;device-type;reserved;reserved;alloc;device-clean
```

たとえば、次の行はデバイス名 `st0:` のエントリを示しています。

```
st0;st;;;;/etc/security/lib/st_clean
```

`device_allocate` 内の各行の終わりに `\` を付けて、エントリを次行に継続させることができます。コメントも挿入できます。`#` を付けると、その後のすべてのテキストは、`\` の直後にない次の改行までコメントになります。どのフィールドでも先行ブランクと後続ブランクを使用できます。

次の各段落では、`device_allocate` ファイル内の各フィールドについて詳しく説明します。

| | |
|---------------------|--|
| <i>device-name</i> | <code>st0</code> 、 <code>fd0</code> 、または <code>sr0</code> . など、デバイス名を指定します。新しい割り当て可能デバイスを作成するときには、 <code>device_maps</code> ファイル内の <i>device-name</i> フィールドから <i>device-name</i> を検索するか、または <code>dminfo</code> コマンドを使用します (この名前は、デバイスの DAC ファイル名でもあります)。 |
| <i>device-type</i> | 汎用デバイスタイプ (<code>st</code> 、 <code>fd</code> 、 <code>sr</code> など、デバイスクラスの名前) を指定します。このフィールドによって、関連するデバイスがグループ化されます。新しい割り当て可能デバイスを作成するときには、 <code>device_maps</code> ファイル内の <i>device-type</i> フィールドから <i>device-type</i> を探るか、または <code>dminfo</code> コマンドを使用してください。 |
| <i>reserved</i> | これらのフィールドは、将来のために予約されています。 |
| <i>alloc</i> | デバイスが割り当て可能かどうかを指定します。このフィールドにアスタリスク (*)が入っている場合は、デバイスが割り当て不可であることを示します。このフィールドに他の文字列が入っている場合や、空の場合は、デバイスが割り当て可能であることを示します。 |
| <i>device-clean</i> | 割り当て処理中にクリーンアップやオブジェクト再使用防止などの特殊処理のために呼び出されるプログラムのパス名を与えます。デバイスが <code>deallocate -F</code> によって強制的に割り当て解除されるときなど、デバイスが <code>deallocate</code> によって有効になると、必ず <i>device-clean</i> プログラムが実行されます。 |

デバイスクリーンスクリプト

デバイスクリーン (*device-clean*) スクリプトは、使用可能なすべてのデータを再使用する前に物理デバイスからパージするというセキュリティ要件に対応するものです。デフォルトでは、カートリッジテープドライブ、フロッピーディスクドライブ、CD-ROM デバイス、オーディオデバイスには、必要なデバイスクリーンスクリプトが用意されています。この節では、デバイスクリーンスクリプトによって実行される処理について説明します。

オブジェクトの再使用

デバイス割り当てによって、オブジェクト再使用の要件の一部が満たされます。デバイスクリーンスクリプトによって、あるユーザがデバイス上に残したデータは、そのデバイスが別のユーザによって割り当て可能にされる前に確実にクリアされます。

テープ用のデバイスクリーンスクリプト

表 4-2 は、サポートされる 3 つのテープデバイスと、それぞれに使用するデバイスクリーンスクリプトを示しています。

表 4-2 サポートされる 3 つのテープデバイスのデバイスクリーンスクリプト

| テープデバイスのタイプ | デバイスクリーンスクリプト |
|--------------------|---------------|
| SCSI 1/4 インチテープ | st_clean |
| アーカイブ 1/4 インチテープ | st_clean |
| オープンリール 1/2 インチテープ | st_clean |

スクリプトは、`mt` の `rewoff1` オプションを使用して、デバイスのクリーンアップに影響を与えます。`mt (1)` マニュアルページを参照してください。スクリプトは、システムブート中に実行されると、デバイスを照会し、そのデバイスがオンラインになっていてメディアが挿入されているかどうかを調べます。

メディアが残っている 1/4 インチのテープデバイスは強制的に割り当てエラー状態になるので、管理者はそのデバイスを手作業でクリーンアップすることになります。

通常のシステム処理中に、`allocate` または `deallocate` を対話型モードで実行すると、割り当てを解除しようとしているデバイスからメディアを取り出すように求めるプロンプトが表示されます。スクリプトは、メディアがデバイスから取り出されるまで一時停止します。

フロッピーディスクと CD-ROM 用のデバイスクリーンスクリプト

表 4-3 は、フロッピーディスクと CD-ROM 用のデバイスクリーンスクリプトを示します。

表 4-3 フロッピーディスクと CD-ROM 用のデバイスクリーンスクリプト

| テープデバイスのタイプ | デバイスクリーンスクリプト |
|-------------|-----------------------|
| フロッピーディスク | <code>fd_clean</code> |
| CD-ROM | <code>sr_clean</code> |

スクリプトは、`eject` コマンドを使用してドライブからメディアを取り出します。`eject(1)` のマニュアルページを参照してください。`eject` が失敗すると、デバイスは割り当てエラー状態になります。

オーディオ用のデバイスクリーンスクリプト

オーディオデバイスは、オーディオクリーンスクリプトによってクリーンアップされます。スクリプトは、`AUDIO_DRAIN ioctl` システムコールを実行してデバイスをフラッシュさせてから、`AUDIO_SETINFO ioctl` システムコールを実行してデバイス構成をデフォルトにリセットします。また、スクリプトは `AUDIOGETREG ioctl` システムコールを使用して、オーディオチップレジスタを検出します。デフォルト以外の値を持つレジスタは、`AUDIOSETREG ioctl` システムコールを使用してリセットされます。

新しいデバイスクリーンスクリプトの作成

システムに新しい割り当て可能デバイスを追加する場合は、独自のデバイスクリーンスクリプトを作成する必要があります。deallocate コマンドは、デバイスクリーンスクリプトにパラメータを渡します。次のように、パラメータはデバイス名が入った文字列です (device_allocate(4) のマニュアルページを参照)。

```
st_clean -[I|F|S] device-name
```

デバイスクリーンスクリプトは、成功した場合は 0 を、失敗した場合は 0 より大きい値を返さなければなりません。オプション -I、-F、-S を使用すると、スクリプトに実行モードを判別させることができます。

-I は、システムブート中にのみ必要です。すべての出力は、システムコンソールに送らなければなりません。失敗した場合や、メディアを強制的に取り出せない場合は、デバイスを割り当てエラー状態にしなければなりません。

-F は強制クリーンアップ用です。このオプションは対話型で、ユーザがプロンプトに回答するものと見なします。このオプションが付いたスクリプトは、クリーンアップの一部に失敗した場合に、クリーンアップ全体を完了しようとしなければなりません。

-S は標準クリーンアップ用です。このオプションは対話型で、ユーザがプロンプトに回答するものと見なします。

ロックファイルの設定

ロックファイルは、/etc/security/dev 内で割り当て可能デバイスごとに 1 つずつ作成される長さ 0 のファイルです。

割り当て可能デバイスのロックファイルがない場合は、そのデバイスを割り当てできず、誰もアクセスできません。

▼ 割り当て可能にするデバイスのロックファイルを設定する方法

1. dminfo コマンドを使用して、デバイスの名前を device_maps ファイル内の該当するエントリから取得します。

65ページの「device_maps ファイル」と、dminfo(1M)、device_maps(4)のマニュアルページを参照してください。たとえば、デバイスタイプ st のデバイス名は st0 です。デバイス名をロックファイル名として使用してください。

2. touch コマンドを使用し、デバイス名を使用してデバイス用の空のロックファイルを作成します。

```
untouchable# cd /etc/security/dev
untouchable# touch device-name
untouchable# chmod 600 device-name
untouchable# chown bin device-name
untouchable# chgrp bin device-name
```

割り当て機構の機能

この節では、割り当て機構の機能について例を挙げて説明します。

allocate コマンドは、まず/etc/security/dev ディレクトリ内で、指定されたデバイスのデバイス名が付いたロックファイルがあるかどうかをチェックします。ファイルが allocate によって所有されている場合は、ロックファイルの所有権が allocate コマンドを入力したユーザの名前に変更されます。

次に、allocate コマンドは device_allocate ファイル内でデバイスのエントリーをチェックし、そのエントリーにデバイスが割り当て可能として設定されているかどうかをチェックします。

次の画面例の最初のリストは、/etc/security/dev 内に、st0 デバイスに使用される所有者 bin、グループ bin、モード 600 のロックファイルがあることを示しています。第 2 のリストは、それに関連するデバイス特殊ファイルが正しく設定されていて、所有者は bin、グループは bin、モードは 000: であることを示しています。

```
untouchable% ls -lg /etc/security/dev/st0
-rw----- 1 bin bin          0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -lg /devices/sbus@1,f8000000/esp@0,800000
c----- 1 bin bin          18,  4 May 12 13:11 st@4,0:
c----- 1 bin bin          18, 20 May 12 13:11 st@4,0:b
c----- 1 bin bin          18, 28 May 12 13:11 st@4,0:bn
c----- 1 bin bin          18, 12 May 12 13:11 st@4,0:c
.
.
c----- 1 bin bin          18,  0 May 12 13:11 st@4,0:u
c----- 1 bin bin          18, 16 May 12 13:11 st@4,0:ub
c----- 1 bin bin          18, 24 May 12 13:11 st@4,0:ubn
c----- 1 bin bin          18,  8 May 12 13:11 st@4,0:un
```

次の画面では、ユーザ `vanessa` がデバイス `st0` を割り当てます。

```
untouchable% whoami
vanessa
untouchable% allocate st0
```

ユーザ `vanessa` が `allocate` コマンドを入力してテープ `st0` を割り当てると、`allocate` はまず `/etc/security/dev/st0` があるかどうかをチェックします。ロックファイルが存在しない場合や、`allocate` 以外のユーザに所有されている場合は、`vanessa` はデバイスを割り当てることができません。

正しい所有権と許可が設定されたロックファイルが見つかると、`allocate` コマンドはそのデバイスのエントリが `device_allocate` ファイルに入っているかどうかと、そのエントリでデバイスが割り当て可能として指定されているかどうかをチェックして確認します。

この例では、`st0` デバイスのデフォルトの `device_allocate` エントリでは、デバイスが割り当て可能として指定されています。`allocate` コマンドでは上記の条件がすべて満たされていることがわかるので、デバイスが `vanessa` に割り当てられます。

`allocate` コマンドは、`/dev` ディレクトリ内でデバイスに関連付けられたデバイス特殊ファイルの所有権と許可を変更します。`st0` デバイスを `vanessa` に割り当てるために、それに関連付けられたデバイス特殊ファイルのモードが `600` に変更され、所有者が `vanessa` に変更されます。

また、`allocate` コマンドは、`/etc/security/dev` ディレクトリ内でデバイスに関連付けられたロックファイルの所有権を変更します。`st0` デバイスを `vanessa` に割り当てるために、`/etc/security/dev/st0` の所有者が `vanessa` に変更されます。

ユーザ `vanessa` がデバイス名 `st0` を使用して `allocate` コマンドを実行すると、次の画面例のように `/etc/security/dev` の所有者が `vanessa` に変更され、それに関連付けられたデバイス特殊ファイルの所有者は `vanessa` になり、`vanessa` はファイルを読み書きする許可を持っていることとなります。

```
untouchable% whoami
vanessa
untouchable% allocate st0
untouchable% ls -lg /etc/security/dev/st0
-rw----- 1 vanessa staff          0 Dec 6 15:21 /etc/security/dev/st0
untouchable% ls -la /devices/sbus@1,f8000000/esp@0,800000
.
.
.
crw----- 1 vanessa 18,  4 May 12 13:11 st@4,0:
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:b
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:bn
```

```
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:c
.
.
.
crw----- 1 vanessa 18, 4 May 12 13:11 st@4,0:u
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:ub
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:ubn
crw----- 1 vanessa 18, 12 May 12 13:11 st@4,0:un
```

デバイスの管理と追加

この節で説明する手順は、デバイスを管理する方法と追加する方法を示しています。

▼ デバイスを管理する方法

1. どのデバイスが `device_allocate` ファイル内でリストされているかと、どのデバイスを割り当て可能にできるかを決定します。
2. どのデバイスを割り当て可能にすべきかを定義します。
3. どのユーザがデバイスを割り当てできるようにするかを決定します。
4. `device_allocate` ファイルを編集して新しいデバイスを追加します。

▼ 新しい割り当て可能デバイスを追加する方法

1. `device_allocate` ファイル内でマシン上の新しい割り当て可能デバイスに関するエントリを作成します。
この方法については、66ページの「`device_allocate` ファイル」を参照してください。
2. `/etc/security/dev` ディレクトリ内で、各割り当て可能デバイスごとに空のロックファイルを作成します。
この方法については、71ページの「ロックファイルの設定」を参照してください。

- 必要であれば、新しいデバイスごとにデバイスクリーンスクリプトを作成します。

Xylogics または Archive テープドライブを追加する場合は、`st_clean` スクリプトを使用できます。それ以外の場合は、独自のスクリプトを作成してください。デバイス処理スクリプトを作成する方法については、69ページの「デバイスクリーンスクリプト」を参照してください。

- デバイスのすべてのデバイス特殊ファイルをユーザ `bin`、グループ `bin`、モード `000` の所有にします。

`dminfo` コマンド実行すると、割り当て可能にしようとするデバイスに関連付けられたすべてのデバイス特殊ファイルの `device_maps` ファイルからリストを取得できます。

デバイス割り当ての使用方法

この節で説明する手順とコマンドは、デバイスを管理する方法と追加する方法を示しています。デバイス割り当てコマンドとデバイス割り当て解除コマンドは、コマンドツールまたはシェルツールウィンドウ内のコマンド行から入力します。

- `allocate` はデバイスをユーザに割り当てます。

表 4-4 のように、2 つの方法のどちらかでデバイスを指定できます。

表 4-4 `allocate` のデバイス指定オプション

| オプション | 動作 |
|-----------------------------|------------------------------|
| <code>device-name</code> | デバイス名と一致するデバイスを割り当てる。 |
| <code>-g device-type</code> | デバイスのグループタイプと一致するデバイスを割り当てる。 |

- `deallocate` は、以前に割り当てられたデバイスを解放します。
- `list_devices` を使用すると、すべての割り当て可能デバイス、現在割り当てられているデバイス、現在は割り当てられていない割り当て可能デバイスのリストを表示できます。

list_devices コマンドには、表 4-5 に示す 3 つのオプションのいずれか 1 つが必要です。

表 4-5 list_devices コマンドのオプション

| オプション | 動作 |
|-------|---|
| -l | すべての割り当て可能デバイスのリスト、またはデバイスに関する情報が表示される。 |
| -n | 現在割り当てられていないデバイスのリスト、またはデバイスに関する情報が表示される。 |
| -u | 現在割り当てられているデバイスのリスト、またはデバイスに関する情報が表示される。 |

▼ デバイスを割り当てる方法

- ◆ 次の例のようにデバイス名を指定して allocate コマンドを使用するか、または -g スイッチを付けて入力します。

```
sar1% allocate st0
```

コマンドでデバイスを割り当てられない場合は、コンソールウィンドウにエラーメッセージが表示されます。全エラーメッセージのリストについては、allocate(1M) のマニュアルページを参照してください。

▼ デバイスの割り当てを解除する方法

- ◆ deallocate コマンドに続けてデバイスファイル名を入力し、テープドライブの割り当てを解除します。

```
sar1% deallocate st0
```

割り当てを解除し終わると、他のユーザがそのデバイスを割り当てることができるようになります。

監査レコードの説明

この付録は 2 部に分かれています。第 1 部では監査レコード構造の各部分と各監査トークンの構造について説明します。第 2 部では、基本セキュリティモジュールによって生成されるすべての監査レコードをイベント記述別に定義します。

- 77ページの「監査レコードの構造」
- 78ページの「監査トークンの構造」
- 96ページの「カーネルレベルで生成される監査レコード」
- 181ページの「ユーザレベルで生成される監査レコード」
- 197ページの「イベントからシステムコールへの変換」

監査レコードの構造

監査レコードは、一連の監査トークンです。監査トークンには、ユーザ ID、時刻、日付などのイベント情報が入っています。header トークンは監査レコードで始まり、省略可能なトレーラで終わります。他の監査トークンには、監査関連情報が入っています。図 A-1 は典型的な監査レコードを示しています。

| |
|--------------|
| header トークン |
| arg トークン |
| data トークン |
| subject トークン |
| return トークン |

図 A-1 典型的な監査レコード

監査トークンの構造

論理上、各トークンにはトークンタイプ識別子とそれに続くトークン固有のデータが付いています。各トークンタイプには固有の形式と構造があります。表 A-1 は現在のトークンを示しています。トークンのスキーマは拡張できます。

表 A-1 基本セキュリティモジュールの監査トークン

| トークン名 | 記述 |
|-----------|---------------------|
| acl | アクセス制御リスト情報 |
| arbitrary | 形式情報と型情報が付いたデータ |
| arg | システムコールの引数値 |
| attr | V ノードトークン |
| exec_args | Exec システムコールの引数 |
| exec_env | Exec システムコールの環境変数 |
| exit | プログラム終了情報 |
| file | 監査ファイル情報 |
| groups | プロセスグループ情報 (使用しません) |
| header | レコードの始まりを示す |

表 A-1 基本セキュリティモジュールの監査トークン 続く

| トークン名 | 記述 |
|-------------|----------------------------------|
| in_addr | インターネットアドレス |
| ip | IP ヘッダ情報 |
| ipc | System V IPC 情報 |
| ipc_perm | System V IPC オブジェクトトークン |
| iport | インターネットポートアドレス |
| newgroups | プロセスグループ情報 |
| opaque | 構造化されていないデータ (形式が未指定) |
| path | パス情報 (パス) |
| process | プロセストークン情報 |
| return | システムコールの状態 |
| seq | シーケンス番号トークン |
| socket | ソケットのタイプとアドレス |
| socket-inet | ソケットのポートとアドレス |
| subject | サブジェクトトークン情報 (process トークンと同じ構造) |
| text | ASCII 文字列 |
| trailer | レコードの終わりを示す |

監査レコードには、必ず header トークンが入っています。header トークンは、監査トレール内で監査レコードの始まりを示します。ユーザの動作に帰因しないイベントからの監査レコードを除き、どの監査レコードにも subject トークンが入っています。ユーザに帰因するイベントの場合、この2つのトークンはイベントを引き起こしたプロセスの値を参照します。非同期イベントの場合、process トークンはシステムを参照します。

acl トークン

acl トークンは ACL に関する情報を記録するもので、4つの固定長フィールドから成ります。このトークンが acl であることを示すトークン ID フィールド、ACL のタイプを表わすフィールド、ACL ID フィールド、およびこの ACL に関連したアクセス権を表わすフィールドです。形式は次のとおりです。

| トークン ID | ACL タイプ | ACL ID | ACL アクセス権 |
|---------|---------|--------|-----------|
| 1 バイト | 4 バイト | 4 バイト | 4 バイト |

図 A-2 acl トークンの形式

arbitrary トークン

arbitrary トークンは、監査トレール用にデータをカプセル化します。このトークンは4つの固定長フィールドと1つのデータ配列からなっています。固定長フィールドは、このトークンを arbitrary トークンとして識別するトークン ID、推奨形式フィールド(16進など)、カプセル化されるデータのサイズを指定するサイズフィールド(短い形式など)、後続の項目数を示すカウントフィールドの4つです。トークンの残りの部分は、指定された型の1つまたは複数の項目からなっています。arbitrary トークンは次のようになっています。

| トークン ID | 出力形式 | 項目サイズ | 項目数 | 項目 1 | ... | 項目 n |
|---------|-------|-------|-------|------|-----|------|
| 1 バイト | 1 バイト | 1 バイト | 1 バイト | | | |

図 A-3 arbitrary トークンの形式

出力形式フィールドには、表 A-2 のような値を入れることができます。

表 A-2 arbitrary トークンの出力形式フィールドの値

| 値 | 動作 |
|-------------|------------------|
| AUP_BINARY | 日付が 2 進形式で出力される |
| AUP_OCTAL | 日付が 8 進形式で出力される |
| AUP_DECIMAL | 日付が 10 進形式で出力される |
| AUP_HEX | 日付が 16 進形式で出力される |
| AUP_STRING | 日付が 10 進形式で出力される |

項目サイズフィールドには、表 A-3 のような値を入れることができます。

表 A-3 arbitrary トークンの項目サイズフィールドの値

| 値 | 動作 |
|-----------|---------------------|
| AUR_BYTE | データはバイト数単位 (1 バイト) |
| AUR_SHORT | データは短い形式の単位 (2 バイト) |
| AUR_LONG | データは長い形式の単位 (4 バイト) |

arg トークン

arg トークンには、システムコールの引数の数、引数の値、省略可能な記述テキスト文字列など、システムコールの引数情報が入っています。このトークンを使用すると、監査レコード内で 32 ビット整数のシステムコール引数を指定できます。arg トークンには 5 つのフィールドがあります。このトークンを arg トークンとして識別するトークン ID、システムコールにトークンの参照先となる引数を指示する引数 ID、引数の値、記述テキスト文字列の長さ、テキスト文字列の 5 つです。図 A-4 はトークンの形式を示しています。

| トークン ID | 引数番号 | 引数の値 | テキスト長 | テキスト |
|---------|-------|-------|-------|--------------|
| 1 バイト | 1 バイト | 4 バイト | 2 バイト | <i>n</i> バイト |

図 A-4 arg トークンの形式

attr トークン

attr トークンには、ファイル *v* ノードからの情報が入っています。このトークンには7つのフィールドがあります。このトークンを attr トークンとして識別するトークン ID、ファイルのアクセスモードとタイプ、所有者ユーザ ID、所有者グループ ID、ファイルシステム ID、*i* ノード ID、ファイルが表すデバイス ID の7つです。ファイルシステム ID とデバイス ID については、statvfs(2) のマニュアルページを参照してください。一般に、このトークンには path トークンが付いており、パスの検索中に生成されます。パス検索エラーが発生すると、必要なファイル情報を取得するために利用できる *v* ノードがないので、このトークンは監査レコードの一部として組み込まれません。図 A-5 は、attr トークンの形式を示しています。

| トークン ID | ファイルモード | 所有者 UID | 所有者 GID | ファイルシステム ID | ファイル <i>i</i> ノード ID | デバイス ID |
|---------|---------|---------|---------|-------------|----------------------|---------|
| 1 バイト | 4 バイト | 4 バイト | 4 バイト | 4 バイト | 4 バイト | 4 バイト |

図 A-5 attr トークンの形式

exec_args トークン

exec_args トークンは、exec システムコールへの引数を記録します。exec_args レコードには、2つの固定長フィールドがあります。一方は、これを exec_args トークンとして識別するトークン ID フィールドです。他方は、exec コールに渡される引数の数を表すカウントフィールドです。トークンの残りの部分は、0 個以上

の NULL で終わる文字列からなっています。図 A-6 は、exec_args トークンを示します。

| トークン ID | カウント | env_args |
|---------|------|----------|
|---------|------|----------|

1 バイト 4 バイト NULL で終わる文字列のカウント

図 A-6 exec_args トークンの形式

注 - exec_args トークンは、監査方針 argv が有効なときにのみ出力されます。詳しくは、40ページの「監査方針の設定」を参照してください。

exec_env トークン

exec_env トークンは、exec システムコールの現在の環境変数を記録します。exec_env レコードには 2 つの固定長フィールドがあります。一方は、これを exec_env トークンとして識別するトークン ID です。他方は、exec コールに渡される引数の数を表すカウントフィールドです。トークンの残りの部分は、0 個以上の NULL で終わる文字列からなっています。図 A-7 は、exec_env トークンを示しています。

| トークン ID | カウント | env_args |
|---------|------|----------|
|---------|------|----------|

1 バイト 4 バイト NULL で終わる文字列のカウント

図 A-7 exec_env トークンの形式

注 - exec_env トークンは、監査方針 arge が有効なときにのみ出力されます。詳しくは、40ページの「監査方針の設定」を参照してください。

exit トークン

exit トークンは、プログラムの終了状態を記録します。exit トークンには、プログラムの終了状態と戻り値が入っています。状態フィールドは exit システムコールに渡されるものと同じです。戻り値フィールドは、システムのエラー番号、または終了状態を詳細に記述する戻り値を示します。図 A-8 は、exit トークンを示しています。

| トークン ID | 状態 | 戻り値 |
|---------|-------|-------|
| 1 バイト | 4 バイト | 4 バイト |

図 A-8 exit トークンの形式

file トークン

file トークンは、新しい監査トレールファイルの始まりと無効になる古いファイルの終りをマークするために、監査デーモンによって生成される特殊なトークンです。監査デーモンは、このトークンが入った特殊な監査レコードを構築して、連続する監査ファイルを1つの監査トレールに「リンク」します。file トークンには4つのフィールドがあります。第1はこれをfile トークンとして識別するトークンID、第2はファイルが作成されるかクローズされた時刻を示す日時のスタンプ、第3はNULLで終わる文字列を含むファイル名のバイト数、第4はNULLで終わる名前が入ったフィールドです。図 A-9 は file トークンを示しています。

| トークン ID | 日時 | 名前の長さ | 前/次のファイル名 |
|---------|-------|-------|--------------|
| 1 バイト | 8 バイト | 2 バイト | <i>n</i> バイト |

図 A-9 file トークンの形式

groups トークン (使用しません)

このトークンは、newgroups トークンに置き換えられています。newgroups トークンは同じタイプの情報をわずかな領域で提供します。ここでは完全を期すために groups トークンについて説明しますが、アプリケーション設計者は newgroups トークンを使用する必要があります。ASCII 形式の出力が表示される時には、どちらのトークン ID にも groups というラベルが付いているため、praudit はこの2つのトークンを区別しないので注意してください。

groups トークンは、プロセスの資格からグループのエントリを記録します。groups トークンには 2 つの固定長フィールドがあります。一方は、これを groups トークンとして識別するトークン ID で、他方はこの監査レコードに入っているグループの数を表すカウントです。図 A-10 は groups トークンを示しています。

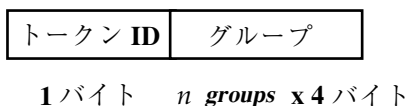


図 A-10 groups トークンの形式

注 - groups トークンは、監査方針 group が有効なときにのみ出力されます。詳しくは、38 ページの「auditconfig コマンド」を参照してください。

header トークン

header トークンは、監査レコードの始まりをマークし、trailer トークンとの組合せでレコード内の他のすべてのトークンを囲むという点で特殊です。header トークンには 6 つのフィールドがあります。これを header トークンとして識別するトークン ID フィールド、ヘッダとトレーラを含めた監査レコードの長さ合計を示すバイト数、監査レコード構造のバージョンを識別するバージョン番号、レコードが表す監査イベントのタイプを識別する監査イベント ID、イベントのタイプに関する補助記述情報が入ったイベント ID 修飾子、レコードの作成日時 の 6 つです。図 A-11 は header トークンを示しています。

| | | | | | |
|---------|---------|---------|---------|--------|-------|
| トークン ID | バイトカウント | バージョン番号 | イベント ID | ID 修飾子 | 日付と時刻 |
| 1 バイト | 4 バイト | 1 バイト | 2 バイト | 2 バイト | 8 バイト |

図 A-11 header トークンの形式

イベント修飾子フィールドでは、次のフラグが定義されています。

```
0x4000  PAD_NOTATTR      nonattributable event
0x8000  PAD_FAILURE      fail audit event
```

in_addr トークン

in_addr トークンには、インターネットアドレスが入っています。この 4 バイト値はインターネットプロトコルアドレスです。このトークンには 2 つのフィールドがあります。一方はこのトークンを in_addr トークンとして識別するトークン ID で、他方はインターネットアドレスです。図 A-12 は in_addr トークンを示しています。

| | |
|---------|-------------|
| トークン ID | インターネットアドレス |
| 1 バイト | 4 バイト |

図 A-12 in_addr トークンの形式

ip トークン

ip トークンには、インターネットプロトコルのヘッダのコピーが入っていますが、IP オプションは含まれていません。IP オプションは、トークン内の IP ヘッダ数を増やせば追加できます。このトークンには 2 つのフィールドがあります。一方はこれを ip トークンとして識別するトークン ID で、他方は IP ヘッダ (すべて 20 バイ

ト)のコピーです。IP ヘッダ構造は、`/usr/include/netinet/ip.h` 内で定義されています。図 A-13は ip トークンを示しています。

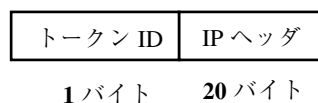


図 A-13 ip トークンの形式

ipc トークン

ipc トークンには、呼び出し元で特定の IPC オブジェクトを識別するための System V IPC メッセージ/セマフォ/共有メモリハンドルが入っています。このトークンには3つのフィールドがあります。第1はこれを ipc トークンとして識別するトークン ID、第2は IPC オブジェクトのタイプを指定するタイプフィールド、第3は IPC オブジェクトを識別するハンドルです。図 A-14は ipc トークンを示しています。

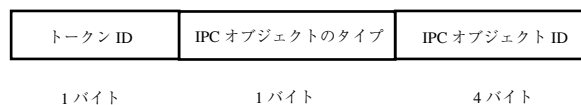


図 A-14 ipc トークンの形式

注 - IPC オブジェクト識別子は Solaris CMW 監査トークンのコンテキストに依存しない性質に違反しています。IPC オブジェクトを一意に識別するグローバルな「名前」はありません。代わりに、IPC オブジェクトが使用可能な間だけ有効なハンドルで識別されます。System V の IPC メカニズムはあまり使用されず、すべてが同じ監査クラスを共有するので、識別は問題ではないはずです。

IPC オブジェクトタイプフィールドには、表 A-4 のような値が入っています。値は `/usr/include/bsm/audit.h` 内で定義されます。

表 A-4 IPC オブジェクトタイプフィールド

| 名前 | 値 | 記述 |
|------------|---|-----------------|
| AU_IPC_MSG | 1 | IPC メッセージオブジェクト |
| AU_IPC_SEM | 2 | IPC セマフォオブジェクト |
| AU_IPC_SHM | 3 | IPC 共有メモリオブジェクト |

ipc_perm トークン

ipc_perm トークンには、System V の IPC アクセス情報が入っています。このトークンは、共有メモリ、セマフォ、メッセージの IPC イベントによって生成された監査レコードに追加されます。このトークンには 8 つのフィールドがあります。具体的には、このトークンを ipc_perm トークンとして識別するトークン ID、IPC 所有者のユーザ ID、IPC 所有者のグループ ID、IPC 作成者のユーザ ID、IPC 作成者のグループ ID、IPC のアクセスモード、IPC のシーケンス番号、IPC キー値の 8 つです。値は、IPC オブジェクトに関連付けられた ipc_perm 構造から取り出されます。図 A-15 は ipc_perm トークンの形式を示しています。

| トークン ID | 所有者 uid | 所有者 gid | 作成者 uid | 作成者 gid | ipc モード | シーケンス ID | IPC 鍵 |
|---------|---------|---------|---------|---------|---------|----------|-------|
| 1 バイト | 4 バイト | 4 バイト | 4 バイト | 4 バイト | 4 バイト | 4 バイト | 4 バイト |

図 A-15 ipc_perm トークンの形式

iport トークン

iport トークンには、TCP (または UDP) ポートアドレスが入っています。このトークンには 2 つのフィールドがあります。一方はこれを iport トークンとして識別するトークン ID で、他方は TCP/UDP ポートアドレスです。図 A-16 は iport トークンを示しています。

| | |
|---------|--------|
| トークン ID | ポート ID |
| 1 バイト | 2 バイト |

図 A-16 ipport トークンの形式

newgroups トークン

このトークンは、groups トークンに代わるものです。ASCII 出力が表示される時には、どちらのトークン ID にも groups というラベルが付いているため、praudit はこの 2 つのトークンを区別しないので注意してください。

newgroups トークンは、プロセスの資格からグループエントリを記録します。newgroups トークンには 2 つの固定長フィールドがあります。一方はこれを newgroups トークンとして識別するトークン ID で、他方はこの監査記録に入っているグループの数を表すカウントです。このトークンの残りの部分は 0 個以上のグループエントリからなっています。図 A-17 は newgroups トークンを示しています。

| | | |
|---------|-------|--------------|
| トークン ID | カウント | グループ |
| 1 バイト | 2 バイト | カウント * 4 バイト |

図 A-17 newgroups トークンの形式

注 - newgroups トークンは、監査方針 group が有効なときにのみ出力されます。詳しくは、38ページの「auditconfig コマンド」を参照してください。

opaque トークン

opaque トークンには、フォーマットされていないデータが一連のバイトとして入っています。このトークンには 3 つのフィールドがあります。これを opaque トークンとして識別するトークン ID、データ量を表すバイト数、バイトデータの配列の 3 つです。図 A-18 は、opaque トークンを示しています。

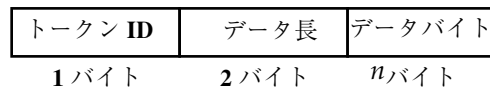


図 A-18 opaque トークンの形式

path トークン

path トークンには、オブジェクトのアクセスパス情報が入っています。このトークンには、トークン ID の他にシステムの実ルートに基づくオブジェクトへの絶対パスが入っています。パスは、パス長を示すバイト数とパスからなっています。図 A-19 は path トークンを示しています。

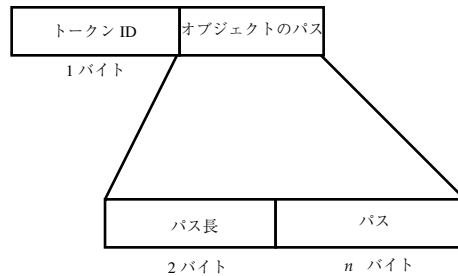


図 A-19 path トークンの形式

process トークン

process トークンには、信号の受信側など、プロセスをオブジェクトとして記述する情報が入っています。このトークンには 9 つのフィールドがあります。このトー

クンを process トークンとして識別するトークン ID、不変の (invariant) 監査 ID、実効ユーザ ID、実効グループ ID、実ユーザ ID、実グループ ID、プロセス ID、監査セッション ID、端末 ID の 9 つです。図 A-20 は process トークンを示しています。

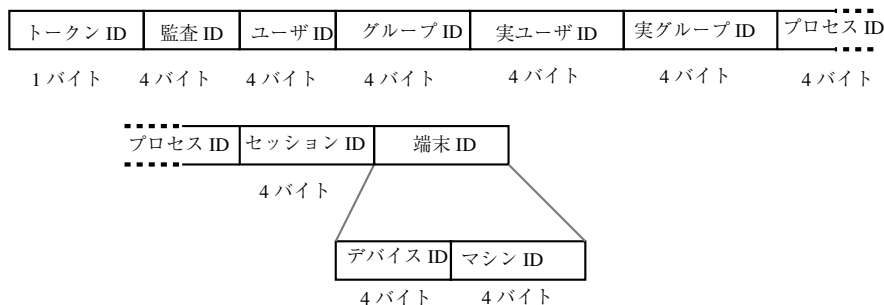


図 A-20 process トークンの形式

監査 ID、ユーザ ID、グループ ID、プロセス ID、端末 ID は、短い形式ではなく長い形式です。

注・セッション ID、実ユーザ ID、または実グループ ID に process トークンのフィールドを使用できないことがあります。その場合、エントリーは -1 に設定されます。

return トークン

return トークンには、システムコールの戻り状態 (u_error) とプロセスの戻り値 (u_rval1) が入っています。このトークンには 3 つのフィールドがあります。第 1 はこのトークンを return トークンとして識別するトークン ID、第 2 はシステムコールのエラー状態、第 3 はシステムコールの戻り値です。このトークンは、必ずシステムコールに関してカーネルによって生成される監査レコードの一部として返されます。このトークンは、アプリケーションを監査中の終了状態と他の戻り値を示します。図 A-21 は return トークンを示しています。

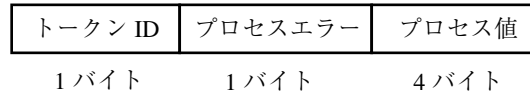


図 A-21 return トークンの形式

seq トークン

seq トークン (シーケンストークン) は、昇順のシーケンス番号が入った省略可能なトークンです。このトークンはデバッグ用です。AUDIT_SEQ 方針が有効になっているときは、各監査レコードにこのトークンが追加されます。seq トークンには 2 つのフィールドがあります。一方はこのトークンを seq トークンとして識別するトークン ID で、他方はシーケンス番号が入った 32 ビットの無符号長形式フィールドです。シーケンス番号は、監査レコードが生成されて監査レコードに組み込まれるたびに 1 ずつ増やされます。図 A-22 は、seq トークンを示しています。

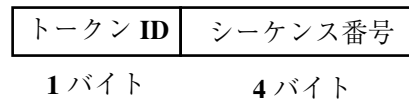


図 A-22 seq トークンの形式

socket トークン

socket トークンには、インターネットソケットを記述する情報が入っています。socket トークンには 6 つのフィールドがあります。つまり、このトークンを socket トークンとして識別するトークン ID、参照されるソケットのタイプ (TCP/UDP/UNIX) を示すソケットタイプフィールド、ローカルポートアドレス、ローカルインターネットアドレス、リモートポートアドレス、リモートインターネットアドレスです。図 A-23 は、socket トークンを示しています。

| | | | | | |
|---------|---------|---------|---------------------|---------|---------------------|
| トークン ID | ソケットタイプ | ローカルポート | ローカルインターネット アドレス | リモートポート | リモートインターネット アドレス |
| 1 バイト | 2 バイト | 2 バイト | 4 バイト | 2 バイト | 4 バイト |

図 A-23 socket トークンの形式

socket-inet トークン

socket-inet トークンは、ローカルポートへのソケット接続を記述します。これは、インターネットの名前空間内でソケット情報を表すために使用されます。socket-inet トークンには 4 つのフィールドがあります。つまり、このトークンを socket-inet トークンとして識別するトークン ID、インターネットファミリー (AF_INET、AF_OSI など) を示すソケットファミリーフィールド、ローカルポートのアドレス、ソケットのアドレスです。図 A-24 は socket-inet トークンを示しています。

| | | | |
|---------|-----------|---------|----------|
| トークン ID | ソケットファミリー | ローカルポート | ソケットアドレス |
| 1 バイト | 2 バイト | 2 バイト | 4 バイト |

図 A-24 socket-inet トークンの形式

subject トークン

subject トークンは、サブジェクト (プロセス) を記述します。構造は process トークンと同じです。このトークンには 9 つのフィールドがあります。つまり、これを subject トークンとして識別する ID、不変の監査 ID、実効ユーザ ID、実効グループ ID、実ユーザ ID、実グループ ID、プロセス ID、監査セッション ID、端末 ID です。このトークンは、必ずシステムコールに関してカーネルによって生成される監査レコードの一部として返されます。図 A-25 は subject トークンを示しています。

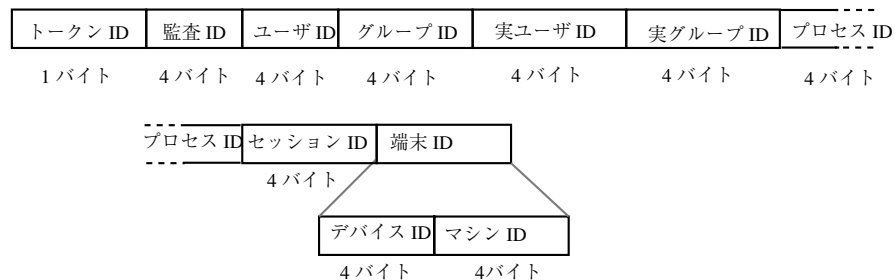


図 A-25 subject トークンの形式

監査 ID、ユーザ ID、グループ ID、プロセス ID、端末 ID は、短い形式ではなく長い形式です。

注 - subject セッション ID、実ユーザ ID、または実グループ ID に subject トークンのフィールドを使用できないことがあります。その場合、エントリーは -1 に設定されます。

text トークン

text トークンにはテキスト文字列が入っています。このトークンには 3 つのフィールドがあります。つまり、このトークンを text トークンとして識別するトークン ID、テキスト文字列の長さ、テキスト文字列そのものです。図 A-26 は text トークンを示しています。

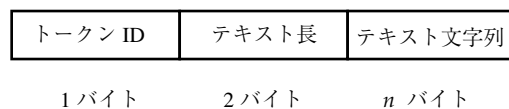


図 A-26 text トークンの形式

trailer トークン

header、trailer という 2 つのトークンは、監査レコードの終端を区別し、他のすべてのトークンを囲むという点で特殊です。trailer トークンは監査レコードを

終了させます。これは省略可能なトークンであり、AUDIT_TRAIL 監査方針が設定されているときにのみ、各レコードの最後のトークンとして追加されます。

trailer トークンは、監査レコードの終端をマークするという点で特殊です。trailer トークンは header トークンとの組合せによって監査レコードを区切ります。また、trailer トークンを使用すると監査トレールを逆方向に検索できます。trailer トークンには、3つのフィールドがあります。つまり、このトークンを trailer トークンとして識別するトークン ID、レコードの終わりをマークしやすくするパッド番号、header トークンと trailer トークンを含めた監査レコード内の合計文字数です。図 A-27 は trailer トークンを示しています。

| トークン ID | パッド番号 | バイト数 |
|---------|-------|-------|
| 1 バイト | 2 バイト | 4 バイト |

図 A-27 trailer トークンの形式

監査トレール分析ソフトウェアによって、各レコードに header と trailer の両方が入ることが保証されます。ファイルシステムがいつぱいのときなど、書き込みエラーが発生すると、監査レコードが不完全になって切り捨てられることがあります。auditsvc は監査トレールへのデータ書き込みを受け持つシステムコールであり、完全な監査レコードを取り出そうとします。auditsvc(2) のマニュアルページを参照してください。ファイルシステムの領域が足りなくなると、コールは現在の監査レコードを解放せずに終了します。コールが再開するときには、切り捨てたレコードを反復できます。

監査レコード

この節では、すべての監査レコードを紹介します。まず、カーネルのイベントによって生成される監査レコードについて説明します(96ページの「カーネルレベルで生成される監査レコード」を参照)。その次に、ユーザーレベルのイベントによって生成される監査レコードについて説明します(181ページの「ユーザーレベルで生成される監査レコード」を参照)。

197ページの「イベントからシステムコールへの変換」には、考えられるすべての監査イベントと、どのカーネルイベントやユーザーイベントによって監査イベントが作成されたかを識別する2つの表が掲載されています。表 A-192 は、監査イベントとシ

システムコールとのマッピングを示しています。表 A-193 は、監査イベントとアプリケーションまたはコマンドとのマッピングを示しています。

一般的な監査レコードの構造

基本セキュリティモジュールによって生成される監査レコードには、一連のトークンが入っています。現在の監査方針に従って、特定のトークンは監査レコード内で省略できます。group、sequence、trailer の各トークンは、いずれもこのカテゴリに該当します。管理者は、auditconfig コマンドの -getpolicy オプションを使用して、これらのトークンが監査レコードに入っているかどうかを判断できます。

カーネルレベルで生成される監査レコード

カーネルに使用されるシステムコールによって、次の監査レコードが作成されます。各レコードは、システムコールのアルファベット順に掲載してあります。各レコードの説明には次の情報が含まれています。

- システムコール名
- 参照先のマニュアルページ (該当する場合)
- 監査イベント番号 (ID)
- 監査イベント名
- 監査イベントクラス
- イベントクラスのマスク
- 監査レコード構造

表 A-5 accept (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|---------|---------|------------|
| AUE_ACCEPT | 33 | nt | 0x00000100 |

形式 (ソケットアドレスが AF_INET ファミリの一部ではない場合) :

header-token

arg-token (1, "fd", file descriptor)

text-token ("bad socket address")

text-token ("bad peer address")

subject-token

return-token

形式 (ソケットアドレスが AF_INET ファミリの一部である場合) :

header-token

以下のファイル記述子に *vnode* がない場合 :

[*arg-token*] (1, "Bad fd", file descriptor)

または、ソケットが結合していない場合 :

[*arg-token*] (1, "fd", file descriptor)

[*text-token*] ("socket not bound")

または、ソケットアドレス長が 0 である場合 :

[*arg-token*] (1, "fd", file descriptor)

[*text-token*] ("bad socket address")

上記以外の場合 :

[*socket-inet-token*] ("socket address")

socket-inet-token ("socket address")

subject-token

return-token

表 A-6 access (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_ACCESS | 14 | fa | 0x00000004 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-7 acl (2) - SETACL コマンド

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_ACLSET | 251 | fm | 0x00000008 |

形式:

header-token

arg-token (2, "cmd", SETACL)

arg-token (3, "nentries", number of ACL entries)

(0..n)[*acl-token*] (ACLs)

subject-token

return-token

表 A-8 acct (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_ACCT | 18 | ad | 0x00000800 |

形式 (zero path) :

header-token

argument-token (1, "accounting off", 0)

subject-token

return-token

形式 (non-zero path) :

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-9 adjtime (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_ADJTIME | 50 | ad | 0x00000800 |

形式 :

header-token

subject-token

return-token

表 A-10 audit (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_AUDIT | 211 | no | 0x00000000 |

形式:

header-token

subject-token

return-token

表 A-11 auditon(2) - get car

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------------|------------|---------|------------|
| AUE_AUDITON_GETCAR | 224 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-12 auditon(2) - get event class

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_AUDITON_GETCLASS | 231 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-13 auditon(2) - get audit state

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------------|------------|---------|------------|
| AUE_AUDITON_GETCOND | 229 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-14 auditon(2) - get cwd

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------------|------------|---------|------------|
| AUE_AUDITON_GETCWD | 223 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-15 auditon(2) - get kernal mask

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_AUDITON_GETKMASK | 221 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-16 auditon(2) - get audit statistics

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------------|------------|---------|------------|
| AUE_AUDITON_GETSTAT | 225 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-17 auditon(2) - GPOLICY command

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------------|------------|---------|------------|
| AUE_AUDITON_GPOLICY | 114 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-18 auditon(2) - GQCTRL command

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------------|------------|---------|------------|
| AUE_AUDITON_GQCTRL | 145 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-19 auditon(2) - set event class

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_AUDITON_SETCLASS | 232 | ad | 0x00000800 |

形式:

header-token

[*argument-token*] (2, "setclass:ec_event", event number)

[*argument-token*] (3, "setclass:ec_class", class mask)

subject-token

return-token

表 A-20 auditon(2) - set audit state

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------------|------------|---------|------------|
| AUE_AUDITON_SETCOND | 230 | ad | 0x00000800 |

形式:

header-token

[*argument-token*] (3, "setcond", audit state)

subject-token

return-token

表 A-21 auditon(2) - set kernal mask

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_AUDITON_SETKMASK | 222 | ad | 0x00000800 |

形式:

header-token

[*argument-token*] (2, "setkmask:as_success", kernel mask)

[*argument-token*] (2, "setkmask:as_failure", kernel mask)

return-token

表 A-22 auditon(2) - set mask per session ID

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_AUDITON_SETSMASK | 228 | ad | 0x00000800 |

形式:

header-token

[*argument-token*] (3, "setsmask:as_success", session ID mask)

[*argument-token*] (3, "setsmask:as_failure", session ID mask)

subject-token

return-token

表 A-23 auditon(2) - reset audit statistics

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------------|------------|---------|------------|
| AUE_AUDITON_SETSTAT | 226 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-24 auditon(2) - set mask per uid

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_AUDITON_SETUMASK | 227 | ad | 0x00000800 |

形式:

header-token

[*argument-token*] (3, "setumask:as_success", audit ID mask)

[*argument-token*] (3, "setumask:as_failure", audit ID mask)

subject-token

return-token

表 A-25 auditon(2) - SPOLICY command

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------------|------------|---------|------------|
| AUE_AUDITON_SPOLICY | 147 | ad | 0x00000800 |

形式:

header-token

[*argument-token*] (1, "policy", audit policy flags)

subject-token

return-token

表 A-26 auditon(2) - SQCTRL command

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------------|------------|---------|------------|
| AUE_AUDITON_SQCTRL | 146 | ad | 0x00000800 |

形式:

header-token

[*argument-token*] (3,"setqctrl:aq_hiwater",queue control param.)

[*argument-token*] (3,"setqctrl:aq_lowater",queue control param.)

[*argument-token*] (3,"setqctrl:aq_bufsz",queue control param.)

[*argument-token*] (3,"setqctrl:aq_delay",queue control param.)

subject-token

return-token

表 A-27 auditsvc(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_AUDITSVC | 136 | ad | 0x00000800 |

形式 (valid file descriptor):

header-token

[*path-token*]

[*attr-token*]

subject-token

return-token

形式 (invalid file descriptor):

header-token

argument-token (1, "no path: fd", fd)

subject-token

return-token

表 A-28 bind(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_BIND | 34 | nt | 0x00000100 |

形式:

header-token

以下のファイル記述子に `vnode` がない場合:

[*arg-token*] (1, "Bad fd", file descriptor)

または、ソケットが `AF_INET` ファミリではない場合:

[*arg-token*] (1, "fd", file descriptor)

text-token] ("bad socket address")

上記以外の場合:

[*arg-token*] (1, "fd", file descriptor)

socket-inet-token] ("socket address")

subject-token

return-token

表 A-29 chdir(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_CHDIR | 8 | pc | 0x00000080 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-30 chmod (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_CHMOD | 10 | fm | 0x00000008 |

形式:

*header-token**argument-token* (2, "new file mode", mode)*path-token*[*attr-token*]*subject-token**return-token*

表 A-31 chown (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_CHOWN | 11 | fm | 0x00000008 |

形式:

*header-token**argument-token* (2, "new file uid", uid)*argument-token* (3, "new file gid", gid)*path-token*[*attr-token*]*subject-token**return-token*

表 A-32 chroot (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_CHROOT | 24 | pc | 0x00000080 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-33 close (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_CLOSE | 112 | c1 | 0x00000040 |

形式:

<file system object>
header-token
argument-token (1, "fd", file descriptor)
 [*path-token*]
 [*attr-token*]
subject-token
return-token

表 A-34 connect (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|---------|---------|------------|
| AUE_CONNECT | 32 | nt | 0x00000100 |

形式 (ソケットアドレスが AF_INET ファミリの一部ではない場合) :

header-token

arg-token (1, "fd", file descriptor)

text-token ("bad socket address")

text-token ("bad peer address")

subject-token

return-token

形式 (ソケットアドレスが AF_INET ファミリの一部である場合) :

header-token

以下のファイル記述子に vnode がない場合 :

[*arg-token*] (1, "Bad fd", file descriptor)

または、ソケットが結合していない場合 :

[*arg-token*] (1, "fd", file descriptor)

text-token ("socket not bound")

または、ソケットアドレス長が 0 である場合 :

[*arg-token*] (1, "fd", file descriptor)

text-token ("bad socket address")

上記以外の場合 :

[*socket-inet-token*] ("socket address")

socket-inet-token ("socket address")

subject-token

return-token

表 A-35 creat (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_CREAT | 4 | fc | 0x00000010 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-36 doorfs (2) - DOOR_BIND

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_BIND | 260 | ip | 0x00000200 |

形式:

header-token

arg-token (1, "door ID", door ID)

subject-token

return-token

表 A-37 doorfs(2) - DOOR_CALL

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_CALL | 254 | ip | 0x00000200 |

形式:

header-token

arg-token (1, "door ID", door ID)

process-token (for process that owns the door)

subject-token

return-token

表 A-38 doorfs(2) - DOOR_CREATE

| イベント名 | イベント ID | イベントクラス | マスク |
|------------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_CREATE | 256 | ip | 0x00000200 |

形式:

header-token

arg-token (1, "door attr", door attributes)

subject-token

return-token

表 A-39 doorfs(2) - DOOR_CRED

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_CRED | 259 | ip | 0x00000200 |

形式:

header-token

subject-token

return-token

表 A-40 doorfs(2) - DOOR_INFO

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_INFO | 258 | ip | 0x00000200 |

形式:

header-token

subject-token

return-token

表 A-41 doorfs(2) - DOOR_RETURN

| イベント名 | イベント ID | イベントクラス | マスク |
|------------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_RETURN | 255 | ip | 0x00000200 |

形式:

header-token

subject-token

return-token

表 A-42 doorfs(2) - DOOR_REVOKE

| イベント名 | イベント ID | イベントクラス | マスク |
|------------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_REVOKE | 257 | ip | 0x00000200 |

形式:

header-token

arg-token (1, "door ID", door ID)

subject-token

return-token

表 A-43 doorfs(2) - DOOR_UNBIND

| イベント名 | イベント ID | イベントクラス | マスク |
|------------------------|------------|---------|------------|
| AUE_DOORFS_DOOR_UNBIND | 261 | ip | 0x00000200 |

形式:

header-token

arg-token (1, "door ID", door ID)

subject-token

return-token

表 A-44 enter prom

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|------------|---------|------------|
| AUE_ENTERPROM | 153 | na | 0x00000400 |

形式:

header-token

text-token (addr, "monitor PROM"|"kadb")

subject-token

return-token

表 A-45 exec(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_EXEC | 7 | pc,ex | 0x40000080 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-46 execve(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_EXECVE | 23 | pc,ex | 0x40000080 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-47 exit prom

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_EXITPROM | 154 | na | 0x00000400 |

形式:

header-token

text-token (addr, "monitor PROM"|"kadb")

subject-token

return-token

表 A-48 exit (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_EXIT | 1 | pc | 0x00000080 |

形式:

header-token

subject-token

return-token

表 A-49 facl(2) - SETACL コマンド

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_FACLSET | 252 | fm | 0x00000008 |

形式 (zero path) :

header-token

arg-token (2, "cmd", SETACL)

arg-token (3, "nentries", number of ACL entries)

arg-token (1, "no path: fd", file descriptor)

(0..n)[*acl-token*] (ACLs)

subject-token

return-token

形式 (non-zero path) :

header-token

arg-token (2, "cmd", SETACL)

arg-token (3, "nentries", number of ACL entries)

path-token

[*attr-token*]

(0..n)[*acl-token*] (ACLs)

subject-token

return-token

表 A-50 fchdir(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_FCHDIR | 68 | pc | 0x00000080 |

形式:

header-token
 [*path-token*]
 [*attr-token*]
subject-token
return-token

表 A-51 fchmod(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_FCHMOD | 39 | fm | 0x00000008 |

形式 (valid file descriptor):

header-token
argument-token (2, "new file mode", mode)
 [*path-token*]
 [*attr-token*]
subject-token
return-token

形式 (invalid file descriptor):

header-token
argument-token (2, "new file mode", mode)
argument-token (1, "no path: fd", fd)
subject-token
return-token

表 A-52 fchown(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_FCHOWN | 38 | fm | 0x00000008 |

形式 (valid file descriptor) :

header-token (2, "new file uid", uid)
argument-token (3, "new file gid", gid)
 [*path-token*]
 [*attr-token*]
subject-token
return-token

形式 (non-file descriptor) :

header-token
argument-token (2, "new file uid", uid)
argument-token (3, "new file gid", gid)
argument-token (1, "no path: fd", fd)
subject-token
return-token

表 A-53 fchroot(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_FCHROOT | 69 | pc | 0x00000080 |

形式 :

header-token
 [*path-token*]
 [*attr-token*]
subject-token
return-token

表 A-54 fcntl (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--|------------|---------|------------|
| AUE_FCNTL (cmd=F_GETLK, F_SETLK, F_SETLKW) | 30 | fm | 0x00000008 |

形式 (file descriptor) :

header-token

argument-token (2, "cmd", cmd)

path-token

attr-token

subject-token

return-token

形式 (bad file descriptor) :

header-token

argument-token (2, "cmd", cmd)

argument-token (1, "no path: fd", fd)

subject-token

return-token

表 A-55 fork(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_FORK | 2 | pc | 0x00000080 |

形式:

header-token

[*argument-token*] (0, "child PID", pid)

subject-token

return-token

監査レコードは子プロセスが生成された時点で生成されるため、fork() の戻り値は不定なので注意する必要がある

表 A-56 fork1(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_FORK1 | 241 | pc | 0x00000080 |

形式:

header-token

[*argument-token*] (0, "child PID", pid)

subject-token

return-token

監査レコードは子プロセスが生成された時点で生成されるため、fork1() の戻り値は不定なので注意する必要がある

表 A-57 fstatfs(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_FSTATFS | 55 | fa | 0x00000004 |

形式 (file descriptor) :

header-token

[*path-token*]

[*attr-token*]

subject-token

return-token

形式 (non-file descriptor) :

header-token

argument-token (1, "no path: fd", fd)

subject-token

return-token

表 A-58 getaudit(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_GETAUDIT | 132 | ad | 0x00000800 |

形式 :

header-token

subject-token

return-token

表 A-59 getauid(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_GETAUID | 130 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-60 getmsg(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_GETMSG | 217 | nt | 0x00000100 |

形式:

header-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

表 A-61 getmsg - accept

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------|------------|---------|------------|
| AUE_SOCKACCEPT | 247 | nt | 0x00000100 |

形式:

header-token

socket-inet-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

表 A-62 getmsg - receive

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_SOCKRECEIVE | 250 | nt | 0x00000100 |

形式:

header-token

socket-inet-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

表 A-63 getpmsg (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_GETPMSG | 219 | nt | 0x00000100 |

形式:

header-token

argument-token (1, "fd", file descriptor)

subject-token

return-token

表 A-64 getportaudit (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------------|------------|---------|------------|
| AUE_GETPORTAUDIT | 149 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-65 inst_sync(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|------------|---------|------------|
| AUE_INST_SYNC | 264 | ad | 0x00000800 |

形式:

header-token

arg-token (2, "flags", flags value)

subject-token

return-token

表 A-66 ioctl(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|---------|---------|------------|
| AUE_IOCTL | 158 | io | 0x20000000 |

形式 (good file descriptor) :

header-token
path-token
 [*attr-token*]
argument-token (2, "cmd" ioctl cmd)
argument-token (3, "arg" ioctl arg)
subject-token
return-token

形式 (socket) :

header-token
 [*socket-token*]
argument-token (2, "cmd" ioctl cmd)
argument-token (3, "arg" ioctl arg)
subject-token
return-token

形式 (non-file file descriptor) :

header-token
argument-token (1, "fd", file descriptor)
argument-token (2, "cmd", ioctl cmd)
argument-token (3, "arg", ioctl arg)
subject-token
return-token

形式 (bad file name) :

header-token
argument-token (1, "no path: fd", fd)
argument-token (2, "cmd", ioctl cmd)
argument-token (3, "arg", ioctl arg)
subject-token
return-token

表 A-67 kill(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_KILL | 15 | pc | 0x00000080 |

形式 (valid process) :

header-token

argument-token (2, "signal", signo)

[*process-token*]

subject-token

return-token

形式 (zero or negative process) :

header-token

argument-token (2, "signal", signo)

argument-token (1, "process", pid)

subject-token

return-token

表 A-68 lchown(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_LCHOWN | 237 | fm | 0x00000008 |

形式:

header-token

argument-token (2, "new file uid", uid)

argument-token (3, "new file gid", gid)

path-token

[*attr-token*]

subject-token

return-token

表 A-69 link(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_LINK | 5 | fc | 0x00000010 |

形式:

header-token

path-token (from path)

[*attr-token*] (from path)

path-token (to path)

subject-token

return-token

表 A-70 lstat (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_LSTAT | 17 | fa | 0x00000004 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-71 lxstat (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_LXSTAT | 236 | fa | 0x00000004 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-72 memcntl(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_MEMCNTL | 238 | ot | 0x80000000 |

形式:

*header-token**argument-token* (1, "base", base address)*argument-token* (2, "len", length)*argument-token* (3, "cmd", command)*argument-token* (4, "arg", command args)*argument-token* (5, "attr", command attributes)*argument-token* (6, "mask", 0)*subject-token**return-token*

表 A-73 mkdir(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_MKDIR | 47 | fc | 0x00000010 |

形式:

*header-token**argument-token* (2, "mode", mode)*path-token*[*attr-token*]*subject-token**return-token*

表 A-74 mknod (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_MKNOD | 9 | fc | 0x00000010 |

形式:

header-token

argument-token (2, "mode", mode)

argument-token (3, "dev", dev)

path-token

[*attr-token*]

subject-token

return-token

表 A-75 mmap(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|---------|---------|------------|
| AUE_MMAP | 210 | no | 0x00000000 |

形式 (valid file descriptor) :

header-token

argument-token (1, "addr", segment address)

argument-token (2, "len", segment length)

[*path-token*]

[*attr-token*]

subject-token

return-token

形式 (invalid file descriptor) :

header-token

argument-token (1, "addr", segment address)

argument-token (2, "len", segment length)

argument-token (1, "no path: fd", fd)

subject-token

return-token

表 A-76 modctl(2) - bind module

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|------------|---------|------------|
| AUE_MODADDMAJ | 246 | ad | 0x00000800 |

形式:

header-token

[*text-token*] (driver major number)

[*text-token*] (driver name)

text-token (root dir.|"no rootdir")

text-token (driver major number|"no drvname")

argument-token (5, "", number of aliases)

(0..n)[*text-token*] (aliases)

subject-token

return-token

表 A-77 modctl(2) - configure module

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|------------|---------|------------|
| AUE_MODCONFIG | 245 | ad | 0x00000800 |

形式:

header-token

text-token (root dir.|"no rootdir")

text-token (driver major number|"no drvname")

subject-token

return-token

表 A-78 modctl(2) - load module

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_MODLOAD | 243 | ad | 0x00000800 |

形式:

header-token

[*text-token*] (default path)

text-token (filename path)

subject-token

return-token

表 A-79 modctl(2) - unload module

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|------------|---------|------------|
| AUE_MODUNLOAD | 244 | ad | 0x00000800 |

形式:

header-token

argument-token (1, "id", module ID)

subject-token

return-token

表 A-80 mount (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_MOUNT | 62 | ad | 0x00000800 |

形式 (UNIX file system) :

header-token

argument-token (3, "flags", flags)

text-token (filesystem type)

path-token

[*attr-token*]

subject-token

return-token

形式 (NFS file system) :

header-token

argument-token (3, "flags", flags)

text-token (filesystem type)

text-token (host name)

argument-token (3, "internal flags", flags)

表 A-81 msgctl(2) - IPC_RMID command

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_MSGCTL_RMID | 85 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

msg ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-82 msgctl(2) - IPC_SET command

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------|------------|---------|------------|
| AUE_MSGCTL_SET | 86 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

msg ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-83 msgctl(2) - IPC_STAT command

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_MSGCTL_STAT | 87 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

msg ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-84 msgget(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_MSGGET | 88 | ip | 0x00000200 |

形式:

header-token

[*ipc-token*]

subject-token

return-token

msg ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-85 msgrcv (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_MSGRCV | 89 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

msg ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-86 msgsnd (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_MSGSND | 90 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "msg ID", message ID)

[*ipc-token*]

subject-token

return-token

msg ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-87 munmap (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_MUNMAP | 214 | c1 | 0x00000040 |

形式:

header-token

argument-token (1, "addr", address of memory)

argument-token (2, "len", memory segment size)

subject-token

return-token

表 A-88 old nice (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_NICE | 203 | pc | 0x00000080 |

形式:

header-token

subject-token

return-token

表 A-89 open(2) - read

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_OPEN_R | 72 | fr | 0x00000001 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-90 open(2) - read, creat

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_OPEN_RC | 73 | fc, fr | 0x00000011 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-91 open(2) - read,creat,trunc

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|----------|------------|
| AUE_OPEN_RTC | 75 | fc,fd,fr | 0x00000031 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-92 open(2) - read,trunc

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_OPEN_RT | 74 | fd,fr | 0x00000021 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-93 open(2) - read,write

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_OPEN_RW | 80 | fr, fw | 0x00000003 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-94 open(2) - read,write,creat

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|------------|------------|
| AUE_OPEN_RWC | 81 | fr, fw, fc | 0x00000013 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-95 open(2) - read,write,create,trunc

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|------------|----------------|------------|
| AUE_OPEN_RWTC | 83 | fr, fw, fc, fd | 0x00000033 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-96 open(2) - read,write,trunc

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|------------|------------|
| AUE_OPEN_RWT | 82 | fr, fw, fd | 0x00000023 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-97 open(2) - write

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_OPEN_W | 76 | fw | 0x00000002 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-98 open(2) - write, creat

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_OPEN_WC | 77 | fw,fc | 0x00000012 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-99 open(2) - write,creat,trunc

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|----------|------------|
| AUE_OPEN_WTC | 79 | fw,fc,fd | 0x00000032 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-100 open(2) - write,trunc

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_OPEN_WT | 78 | fw,fd | 0x00000022 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-101 p_online(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------------|---|---------|------------|
| AUE_P_ONLINE | 262 | ad | 0x00000800 |
| <i>header-token</i> | | | |
| <i>arg-token</i> | (1, "processor ID", processor ID) | | |
| <i>arg-token</i> | (2, "flags", flags value) | | |
| <i>text-token</i> | (text form of flags value: P_ONLINE, P_OFFLINE, P_STATUS) | | |
| <i>subject-token</i> | | | |
| <i>return-token</i> | | | |

表 A-102 pathconf(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------------|------------|---------|------------|
| AUE_PATHCONF | 71 | fa | 0x00000004 |
| 形式: | | | |
| <i>header-token</i> | | | |
| <i>path-token</i> | | | |
| [<i>attr-token</i>] | | | |
| <i>subject-token</i> | | | |
| <i>return-token</i> | | | |

表 A-103 pipe(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_PIPE | 185 | no | 0x00000000 |

形式:

header-token

subject-token

return-token

表 A-104 priocntlsys(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_PRIOCNTLSYS | 212 | pc | 0x00000080 |

形式:

header-token

argument-token (1, "pc_version", priocntl version num.)

argument-token (3, "cmd", command)

subject-token

return-token

表 A-105 process dumped core

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|-----------|
| AUE_CORE | 111 | fc | 0x0000010 |

形式:

header-token

path-token

[*attr-token*]

argument-token (1, "signal", signal)

subject-token

return-token

表 A-106 processor_bind(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------------|------------|---------|------------|
| AUE_PROCESSOR_BIND | 263 | ad | 0x00000800 |

形式 (プロセッサ結合なし) :

header-token

arg-token (1, "ID type", type of ID)

arg-token (2, "ID", ID value)

text-token ("PBIND_NONE")

process-token (for process whose threads are bound to the processor)

subject-token

return-token

形式 (プロセッサ結合あり) :

header-token

arg-token (1, "ID type", type of ID)

arg-token (2, "ID", ID value)

arg-token (3, "processor ID", processor ID)

process-token (for process whose threads are bound to the processor)

subject-token

return-token

表 A-107 putmsg(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_PUTMSG | 216 | nt | 0x00000100 |

形式:

header-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

表 A-108 putmsg-connect

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_SOCKCONNECT | 248 | nt | 0x00000100 |

形式:

header-token

socket-inet-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

表 A-109 putmsg-send

| イベント名 | イベント ID | EventClass | マスク |
|----------------|------------|------------|------------|
| AUE_SOCKETSEND | 249 | nt | 0x00000100 |

形式:

header-token

socket-inet-token

argument-token (1, "fd", file descriptor)

argument-token (4, "pri", priority)

subject-token

return-token

表 A-110 putpmsg(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_PUTPMSG | 218 | nt | 0x00000100 |

形式:

header-token

argument-token (1, "fd", file descriptor)

subject-token

return-token

表 A-111 readlink(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_READLINK | 22 | fr | 0x00000001 |

形式:

*header-token**path-token**[attr-token]**subject-token**return-token*

表 A-112 rename(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_RENAME | 42 | fc,fd | 0x00000030 |

形式:

*header-token**path-token* (from name)*[attr-token]* (from name)*[path-token]* (to name)*subject-token**return-token*

表 A-113 rmdir(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_RMDIR | 48 | fd | 0x00000020 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-114 semctl(2) - getall

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------------|------------|---------|------------|
| AUE_SEMCTL_GETALL | 105 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-115 semctl(2) - GETNCNT command

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------------|------------|---------|------------|
| AUE_SEMCTL_GETNCNT | 102 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-116 semctl(2) - GETPID command

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------------|------------|---------|------------|
| AUE_SEMCTL_GETPID | 103 | ip | 0x00000200 |

形式:

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-117 semctl(2) - GETVAL command

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------------|---------|---------|------------|
| AUE_SEMCTL_GETVAL | 104 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-118 semctl(2) - GETZCNT command

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------------|---------|---------|------------|
| AUE_SEMCTL_GETZCNT | 106 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-119 semctl(2) - IPC_RMID command

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_SEMCTL_RMID | 99 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-120 semctl(2) - IPC_SET command

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------|------------|---------|------------|
| AUE_SEMCTL_SET | 100 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-121 semctl(2) - SETALL command

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------------|---------|---------|------------|
| AUE_SEMCTL_SETALL | 108 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-122 semctl(2) - SETVAL command

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------------|---------|---------|------------|
| AUE_SEMCTL_SETVAL | 107 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-123 semctl(2) - IPC_STAT command

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_SEMCTL_STAT | 101 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

表 A-124 semget(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_SEMGET | 109 | ip | 0x00000200 |

形式:

header-token

[*ipc-token*]

subject-token

return-token

システムコールが失敗した場合、ipc トークンと ipc_perm トークンは含まれない

表 A-125 semop (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_SEMOP | 110 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "sem ID", semaphore ID)

[*ipc-token*]

subject-token

return-token

セマフォ ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-126 setaudit(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_SETAUDIT | 133 | ad | 0x00000800 |

形式 (有効なプログラムスタックのアドレス):

header-token

argument-token (1, "setaudit:audit", audit user ID)

argument-token (1, "setaudit:port", terminal ID)

argument-token (1, "setaudit:machine", terminal ID)

argument-token (1, "setaudit:as_success", preselection mask)

argument-token (1, "setaudit:as_failure", preselection mask)

argument-token (1, "setaudit:asid", audit session ID)

subject-token

return-token

形式 (無効なプログラムスタックのアドレス):

header-token

subject-token

return-token

表 A-127 setaudit(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_SETAUDIT | 131 | ad | 0x00000800 |

形式:

header-token

argument-token (2, "setaudit", audit user ID)

subject-token

return-token

表 A-128 setegid(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_SETEGID | 214 | pc | 0x00000080 |

形式:

header-token

argument-token (1, "gid", group ID)

subject-token

return-token

表 A-129 seteuid(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_SETEUID | 215 | pc | 0x00000080 |

形式:

header-token

argument-token (1, "gid", user ID)

subject-token

return-token

表 A-130 old setgid(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|---------|---------|------------|
| AUE_SETGID | 205 | pc | 0x00000080 |

形式:

header-token

argument-token (1, "gid", group ID)

subject-token

return-token

表 A-131 setgroups(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|---------|---------|------------|
| AUE_SETGROUPS | 26 | pc | 0x00000080 |

形式:

header-token

[*argument-token*] (1, "setgroups", group ID)

subject-token

return-token

グループセットごとに1つずつトークンがある

表 A-132 setpgrp(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_SETPGRP | 27 | pc | 0x00000080 |

形式:

header-token

subject-token

return-token

表 A-133 setregid(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_SETREGID | 41 | pc | 0x00000080 |

形式:

header-token

arg-token (1, "rgid", real group ID)

arg-token (2, "egid", effective group ID)

subject-token

return-token

表 A-134 setreuid(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|------------|---------|------------|
| AUE_SETREUID | 40 | pc | 0x00000080 |

形式:

header-token

arg-token (1, "ruid", real user ID)

arg-token (2, "euid", effective user ID)

subject-token

return-token

表 A-135 setrlimit(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|---------------|------------|---------|------------|
| AUE_SETRLIMIT | 51 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-136 old setuid(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|---------|---------|------------|
| AUE_OSETUID | 200 | pc | 0x00000080 |

形式:

header-token

argument-token (1, "uid", user ID)

subject-token

return-token

監査ソフトウェアに現在含まれているバグの関係で、このトークンは AUE_OSETUID として表示される

表 A-137 shmat (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|---------|---------|------------|
| AUE_SHMAT | 96 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "shmid", shared memory ID)

argument-token (2, "shmaddr", shared mem addr)

[*ipc-token*]

[*ipc_perm-token*]

subject-token

return-token

共有メモリのセグメント ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-138 shmctl(2) - IPC_RMID command

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_SHMCTL_RMID | 92 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "shmctl", shared memory ID)

[*ipc-token*]

subject-token

return-token

共有メモリのセグメント ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-139 shmctl(2) - IPC_SET command

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------|------------|---------|------------|
| AUE_SHMCTL_SET | 93 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "shmctl", shared memory ID)

[*ipc-token*]

[*ipc_perm-token*]

subject-token

return-token

共有メモリのセグメント ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-140 shmctl(2) - IPC_STAT command

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------------|------------|---------|------------|
| AUE_SHMCTL_STAT | 94 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "shmid", shared memory ID)

[*ipc-token*]

subject-token

return-token

共有メモリのセグメント ID が無効な場合、ipc トークンと ipc_perm トークンは含まれない

表 A-141 shmdt(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_SHMDT | 97 | ip | 0x00000200 |

形式:

header-token

argument-token (1, "shmaddr", shared mem addr)

subject-token

return-token

表 A-142 shmget (2)

| イベント名 | イベント ID | EventClass | マスク |
|------------|------------|------------|------------|
| AUE_SHMGET | 95 | ip | 0x00000200 |

形式:

header-token

argument-token (0, "shmID", shared memory ID)

[*ipc-token*]

[*ipc_perm-token*]

subject-token

return-token

失敗したイベントの場合、ipc トークンと ipc_perm トークンは含まれない

表 A-143 shutdown(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|--------------|---------|---------|------------|
| AUE_SHUTDOWN | 46 | nt | 0x00000100 |

形式 (ソケットアドレスが AF_INET ファミリの一部ではない場合) :

header-token

arg-token (1, "fd", file descriptor)

text-token ["bad socket address"]

text-token ["bad peer address"]

subject-token

return-token

形式 (ソケットアドレスが AF_INET ファミリの一部である場合) :

header-token

以下のファイル記述子に *vnode* がない場合 :

[arg-token] (1, "Bad fd", file descriptor)

または、ソケットが結合していない場合 :

[arg-token] (1, "fd", file descriptor)

text-token ["socket not bound"]

または、ソケットアドレス長が 0 である場合 :

[arg-token] (1, "fd", file descriptor)

text-token ["bad socket address"]

上記以外の場合 :

[socket-inet-token] ("socket address")

socket-inet-token ("socket address")

subject-token

return-token

表 A-144 stat(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|----------|------------|---------|------------|
| AUE_STAT | 16 | fa | 0x00000004 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-145 statfs(2)

| イベント名 | イベント ID | EventClass | マスク |
|------------|------------|------------|------------|
| AUE_STATFS | 54 | fa | 0x00000004 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-146 statvfs(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_STATVFS | 234 | fa | 0x00000004 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-147 stime(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_STIME | 201 | ad | 0x00000800 |

形式:

header-token
subject-token
return-token

表 A-148 symlink(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_SYMLINK | 21 | fc | 0x00000010 |

形式:

*header-token**text-token* (symbolic link string)*path-token*[*attr-token*]*subject-token**return-token*

表 A-149 sysinfo(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-------------|------------|---------|------------|
| AUE_SYSINFO | 39 | ad | 0x00000800 |

形式:

*header-token**argument-token* (1, "cmd", command)*text-token* (name)*subject-token**return-token*

表 A-150 system booted

| イベント名 | イベント ID | イベントクラス | マスク |
|----------------|------------|---------|------------|
| AUE_SYSTEMBOOT | 113 | na | 0x00000400 |

形式:

header-token

text-token ("booting kernel")

return-token

表 A-151 umount(2) - old version

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_UMOUNT | 12 | ad | 0x00000800 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-152 unlink(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_UNLINK | 6 | fd | 0x00000020 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-153 old utime(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_UTIME | 202 | fm | 0x00000008 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-154 utimes(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_UTIMES | 49 | fm | 0x00000008 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-155 utssys(2) - fusers

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_UTSSYS | 233 | ad | 0x00000800 |

形式:

header-token
path-token
 [*attr-token*]
subject-token
return-token

表 A-156 vfork(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_VFORK | 25 | pc | 0x00000080 |

形式:

header-token

argument-token (0, "child PID", pid)

subject-token

return-token

監査レコードは子プロセスが作成された時点で作成されるため、fork の戻り値は不定なので注意する必要がある

表 A-157 vtrace(2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_VTRACE | 36 | pc | 0x00000080 |

形式:

header-token

subject-token

return-token

表 A-158 xmknod (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|------------|------------|---------|------------|
| AUE_XMKNOD | 240 | fc | 0x00000010 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

表 A-159 xstat (2)

| イベント名 | イベント ID | イベントクラス | マスク |
|-----------|------------|---------|------------|
| AUE_XSTAT | 235 | fa | 0x00000004 |

形式:

header-token

path-token

[*attr-token*]

subject-token

return-token

ユーザレベルで生成される監査レコード

カーネルの外側で動作するアプリケーションによって、次の監査レコードが作成されます。各レコードはプログラムのアルファベット順に掲載されています。各レコードの説明には、次の情報が含まれています。

- プログラム名

- 参照先のマニュアルページ (該当する場合)
- 監査イベント番号
- 監査イベント名
- 監査レコードの構造

表 A-160 allocate-device success

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-------------------|--------------------|---------|----------|------------|
| AUE_allocate_succ | /usr/sbin/allocate | 6200 | ad | 0x00000800 |

形式:

header-token

subject-token

newgroups-token

exit-token

表 A-161 allocate-device failure

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-------------------|--------------------|---------|----------|------------|
| AUE_allocate_fail | /usr/sbin/allocate | 6201 | ad | 0x00000800 |

形式:

header-token

subject-token

newgroups-token

exit-token

表 A-162 deallocate-device success

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-------------------|--------------------|------------|-------------|------------|
| AUE_allocate_succ | /usr/sbin/allocate | 6202 | ad | 0x00000800 |

形式:

header-token

subject-token

newgroups-token

exit-token

表 A-163 deallocate-device failure

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-------------------|--------------------|------------|-------------|------------|
| AUE_allocate_fail | /usr/sbin/allocate | 6203 | ad | 0x00000800 |

形式:

header-token

subject-token

newgroups-token

exit-token

表 A-164 allocate-list devices success

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|---------------------|--------------------|------------|-------------|------------|
| AUE_listdevice_succ | /usr/sbin/allocate | 6205 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-165 allocate-list devices failure

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|---------------------|------------------------|------------|-------------|------------|
| AUE_listdevice_fail | /usr/sbin/ allocate | 6206 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-166 at-create crontab

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|---------------|-------------|------------|-------------|------------|
| AUE_at_create | /usr/bin/at | 6144 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-167 at-delete atjob (at or atrm)

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|---------------|-------------|------------|-------------|------------|
| AUE_at_delete | /usr/bin/at | 6145 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-168 at-permission

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-------------|-------------|------------|-------------|------------|
| AUE_at_perm | /usr/bin/at | 6146 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-169 crontab-crontab created

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|--------------------|------------------|------------|-------------|------------|
| AUE_crontab_create | /usr/bin/crontab | 6148 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-170 crontab-crontab deleted

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|--------------------|------------------|------------|-------------|------------|
| AUE_crontab_delete | /usr/bin/crontab | 6149 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-171 cron-invoke atjob or crontab

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-----------------|------------------|------------|-------------|------------|
| AUE_cron_invoke | /usr/bin/crontab | 6147 | ad | 0x00000800 |

形式:

header-token

subject-token

text-token (program)

text-token (shell)

text-token (cmd)

exit-token

表 A-172 crontab-permission

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------------|------------------|------------|-------------|------------|
| AUE_crontab_perm | /usr/bin/crontab | 6150 | ad | 0x00000800 |

形式:

header-token

subject-token

[*group-token*]

exit-token

表 A-173 halt(1m)

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------------|----------------|------------|-------------|------------|
| AUE_halt_solaris | /usr/sbin/halt | 6160 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-174 inetd

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-------------------|-----------------|------------|-------------|------------|
| AUE_inetd_connect | /usr/sbin/inetd | 6151 | na | 0x00000400 |

形式:

header-token

subject-token

text-token (service name)

return-token

表 A-175 init (1m)

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------------|--|------------|-------------|------------|
| AUE_init_solaris | /sbin/init; /usr/ sbin/init; /usr/ sbin/shutdown | 6166 | ad | 0x00000800 |

形式:

header-token

subject-token

text-token (init level)

return-token

表 A-176 ftp access

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|---------|-------------------|------------|-------------|------------|
| AUE_ftp | /usr/sbin/in.ftpd | 6165 | 10 | 0x00001000 |

形式:

header-token

subject-token

text-token (error message, failure only)

return-token

表 A-177 login - local

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-----------|-----------------|------------|-------------|------------|
| AUE_login | /usr/sbin/login | 6152 | 10 | 0x00001000 |

形式:

header-token

subject-token

text-token (error message)

return-token

表 A-178 login - rlogin

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------|-----------------|------------|-------------|------------|
| AUE_rlogin | /usr/sbin/login | 6155 | lo | 0x00001000 |

形式:

header-token

subject-token

text-token (error message)

return-token

表 A-179 login - telnet

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------|-----------------|------------|-------------|------------|
| AUE_telnet | /usr/sbin/login | 6154 | lo | 0x00001000 |

形式:

header-token

subject-token

text-token (error message)

return-token

表 A-180 logout

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------|-----------------|------------|-------------|------------|
| AUE_logout | /usr/sbin/login | 6153 | lo | 0x00001000 |

形式:

*header-token**subject-token**return-token*

表 A-181 mount

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------------|---------------------|------------|-------------|------------|
| AUE_mountd_mount | /usr/lib/nfs/mountd | 6156 | na | 0x00000400 |

形式:

*header-token**subject-token**text-token* (remote client hostname)*path-token* (mount dir)*text-token* (error message, failure only)*return-token*

表 A-182 unmount

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|-------------------|---------------------|------------|-------------|------------|
| AUE_mountd_umount | /usr/lib/nfs/mountd | 6157 | na | 0x00000400 |

形式:

header-token

subject-token

text-token (remote client hostname)

path-token (mount dir)

text-token (error message, failure only)

return-token

表 A-183 passwd

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|---------------|-----------------|------------|-------------|------------|
| AUE_passwoeod | /usr/bin/passwd | 6163 | lo | 0x00001000 |

形式:

header-token

subject-token

text-token (error message)

return-token

表 A-184 poweroff(1m)

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|--------------------------|--------------------|------------|-------------|------------|
| AUE_poweroff _solaris | /usr/sbin/poweroff | 6169 | ad | 0x00000800 |

形式:
header-token
subject-token
return-token

表 A-185 reboot (1m)

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|--------------------|------------------|------------|-------------|------------|
| AUE_reboot_solaris | /usr/sbin/reboot | 6161 | ad | 0x00000800 |

形式:
header-token
subject-token
return-token

表 A-186 rexd

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|----------|--------------------|------------|-------------|------------|
| AUE_rexd | /usr/sbin/rpc.rexd | 6164 | 1o | 0x00001000 |

形式:

header-token

subject-token

text-token (error message, failure only)

text-token (hostname)

text-token (username)

text-token (command to be executed)

exit-token

表 A-187 rexecd

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|------------|---------------------|------------|-------------|------------|
| AUE_rexecd | /usr/sbin/in.rexecd | 6162 | 1o | 0x00001000 |

形式:

header-token

subject-token

text-token (error message, failure only)

text-token (hostname)

text-token (username)

text-token (command to be executed)

exit-token

表 A-188 rsh access

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|----------|-------------------|------------|-------------|------------|
| AUE_rshd | /usr/sbin/in.rshd | 6158 | lo | 0x00001000 |

形式:

header-token

subject-token

text-token (command string)

text-token (local user)

text-token (remote user)

return-token

表 A-189 shutdown(1b)

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|--------------------------|-------------------|------------|-------------|------------|
| AUE_shutdown _solaris | /usr/ucb/shutdown | 6168 | ad | 0x00000800 |

形式:

header-token

subject-token

return-token

表 A-190 su

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|--------|-------------|---------|----------|------------|
| AUE_su | /usr/bin/su | 6159 | lo | 0x00001000 |

形式:

header-token

text-token (error message)

subject-token

return-token

表 A-191 uadmin(1m)

| イベント名 | プログラム | イベント ID | イベント クラス | マスク |
|--------------------|--------------------------------|---------|----------|------------|
| AUE_uadmin_solaris | /sbin/uadmin; /usr/sbin/uadmin | 6167 | ad | 0x00000800 |

形式:

header-token

subject-token

text-token (function)

text-token (argument)

return-token

イベントからシステムコールへの変換

表 A-192 は、監査イベント名とそれを作成したシステムコールまたはカーネルイベントとの関連付けを示しています。表 A-193 は、監査イベントとそれを生成したアプリケーションまたはコマンドとの関連付けを示しています。

表 A-192 イベントからシステムコールへの変換

| 監査イベント | システムコール |
|----------------------|---------|
| AUE_ACCEPT | 表 A-5 |
| AUE_ACCESS | 表 A-6 |
| AUE_ACLSET | 表 A-7 |
| AUE_ACCT | 表 A-8 |
| AUE_ADJTIME | 表 A-9 |
| AUE_AUDIT | 表 A-10 |
| AUE_AUDITON_GETCAR | 表 A-11 |
| AUE_AUDITON_GETCLASS | 表 A-12 |
| AUE_AUDITON_GETCOND | 表 A-13 |
| AUE_AUDITON_GETCWD | 表 A-14 |
| AUE_AUDITON_GETKMASK | 表 A-15 |
| AUE_AUDITON_GETSTAT | 表 A-16 |
| AUE_AUDITON_GPOLICY | 表 A-17 |
| AUE_AUDITON_GQCTRL | 表 A-18 |
| AUE_AUDITON_SETCLASS | 表 A-19 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|----------------------|---------|
| AUE_AUDITON_SETCOND | 表 A-20 |
| AUE_AUDITON_SETKMASK | 表 A-21 |
| AUE_AUDITON_SETSMASK | 表 A-22 |
| AUE_AUDITON_SETSTAT | 表 A-23 |
| AUE_AUDITON_SETUMASK | 表 A-24 |
| AUE_AUDITON_SPOLICY | 表 A-25 |
| AUE_AUDITON_SQCTRL | 表 A-26 |
| AUE_AUDITSVC | 表 A-27 |
| AUE_BIND | 表 A-28 |
| AUE_CHDIR | 表 A-29 |
| AUE_CHMOD | 表 A-30 |
| AUE_CHOWN | 表 A-31 |
| AUE_CHROOT | 表 A-32 |
| AUE_CLOSE | 表 A-33 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|------------------------|---------|
| AUE_CONNECT | 表 A-34 |
| AUE_CORE | 表 A-105 |
| AUE_CREAT | 表 A-35 |
| AUE_DOORFS_DOOR_BIND | 表 A-36 |
| AUE_DOORFS_DOOR_CALL | 表 A-37 |
| AUE_DOORFS_DOOR_CREATE | 表 A-38 |
| AUE_DOORFS_DOOR_CRED | 表 A-39 |
| AUE_DOORFS_DOOR_INFO | 表 A-40 |
| AUE_DOORFS_DOOR_RETURN | 表 A-41 |
| AUE_DOORFS_DOOR_REVOKE | 表 A-42 |
| AUE_DOORFS_DOOR_UNBIND | 表 A-43 |
| AUE_ENTERPROM | 表 A-44 |
| AUE_EXEC | 表 A-45 |
| AUE_EXECVE | 表 A-46 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|--------------|---------|
| AUE_EXIT | 表 A-48 |
| AUE_EXITPROM | 表 A-47 |
| AUE_FACLSET | 表 A-49 |
| AUE_FCHDIR | 表 A-50 |
| AUE_FCHMOD | 表 A-51 |
| AUE_FCHOWN | 表 A-52 |
| AUE_FCHROOT | 表 A-53 |
| AUE_FCNTL | 表 A-54 |
| AUE_FORK | 表 A-55 |
| AUE_FORK1 | 表 A-56 |
| AUE_FSTATFS | 表 A-57 |
| AUE_GETAUDIT | 表 A-58 |
| AUE_GETAUID | 表 A-59 |
| AUE_GETMSG | 表 A-60 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|------------------|---------|
| AUE_GETPMSG | 表 A-63 |
| AUE_GETPORTAUDIT | 表 A-64 |
| AUE_INST_SYNC | 表 A-65 |
| AUE_IOCTL | 表 A-66 |
| AUE_KILL | 表 A-67 |
| AUE_LCHOWN | 表 A-68 |
| AUE_LINK | 表 A-69 |
| AUE_LSTAT | 表 A-70 |
| AUE_LXSTAT | 表 A-71 |
| AUE_MEMCNTL | 表 A-72 |
| AUE_MKDIR | 表 A-73 |
| AUE_MKNOD | 表 A-74 |
| AUE_MMAP | 表 A-75 |
| AUE_MODADDMAJ | 表 A-76 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|-----------------|---------|
| AUE_MODCONFIG | 表 A-77 |
| AUE_MODLOAD | 表 A-78 |
| AUE_MODUNLOAD | 表 A-79 |
| AUE_MOUNT | 表 A-80 |
| AUE_MSGCTL_RMID | 表 A-81 |
| AUE_MSGCTL_SET | 表 A-82 |
| AUE_MSGCTL_STAT | 表 A-83 |
| AUE_MSGGET | 表 A-84 |
| AUE_MSGRCV | 表 A-85 |
| AUE_MSGSND | 表 A-86 |
| AUE_MUNMAP | 表 A-87 |
| AUE_NICE | 表 A-88 |
| AUE_OPEN_R | 表 A-89 |
| AUE_OPEN_RC | 表 A-90 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|---------------|---------|
| AUE_OPEN_RT | 表 A-92 |
| AUE_OPEN_RTC | 表 A-91 |
| AUE_OPEN_RW | 表 A-93 |
| AUE_OPEN_RWC | 表 A-94 |
| AUE_OPEN_RWT | 表 A-96 |
| AUE_OPEN_RWTC | 表 A-95 |
| AUE_OPEN_W | 表 A-97 |
| AUE_OPEN_WC | 表 A-98 |
| AUE_OPEN_WT | 表 A-100 |
| AUE_OPEN_WTC | 表 A-99 |
| AUE_OSETUID | 表 A-136 |
| AUE_P_ONLINE | 表 A-101 |
| AUE_PATHCONF | 表 A-102 |
| AUE_PIPE | 表 A-103 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|--------------------|---------|
| AUE_PRIOCNTLSYS | 表 A-104 |
| AUE_PROCESSOR_BIND | 表 A-106 |
| AUE_PUTMSG | 表 A-107 |
| AUE_PUTPMSG | 表 A-110 |
| AUE_READLINK | 表 A-111 |
| AUE_RENAME | 表 A-112 |
| AUE_RMDIR | 表 A-113 |
| AUE_SEMCTL_GETALL | 表 A-114 |
| AUE_SEMCTL_GETNCNT | 表 A-115 |
| AUE_SEMCTL_GETPID | 表 A-116 |
| AUE_SEMCTL_GETVAL | 表 A-117 |
| AUE_SEMCTL_GETZCNT | 表 A-118 |
| AUE_SEMCTL_RMID | 表 A-119 |
| AUE_SEMCTL_SET | 表 A-120 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|-------------------|---------|
| AUE_SEMCTL_SETALL | 表 A-121 |
| AUE_SEMCTL_SETVAL | 表 A-122 |
| AUE_SEMCTL_STAT | 表 A-123 |
| AUE_SEMGET | 表 A-124 |
| AUE_SEMOP | 表 A-125 |
| AUE_SETAUDIT | 表 A-126 |
| AUE_SETAUID | 表 A-127 |
| AUE_SETEGID | 表 A-128 |
| AUE_SETEUID | 表 A-129 |
| AUE_SETGID | 表 A-130 |
| AUE_SETGROUPS | 表 A-131 |
| AUE_SETPGRP | 表 A-132 |
| AUE_SETREGID | 表 A-133 |
| AUE_SETREUID | 表 A-134 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|-------------------|---------|
| AUE_SETRLIMIT | 表 A-135 |
| AUE_SETTUID | 表 A-136 |
| AUE_SHMAT | 表 A-137 |
| AUE_SHMCTL_RMID | 表 A-138 |
| AUE_SHMCTL_SET | 表 A-139 |
| AUE_SHMCTL_STAT | 表 A-140 |
| AUE_SHMDT | 表 A-141 |
| AUE_SHMGET | 表 A-142 |
| AUE_SHUTDOWN | 表 A-143 |
| AUE_SOCKETACCEPT | 表 A-61 |
| AUE_SOCKETCONNECT | 表 A-108 |
| AUE_SOCKETRECEIVE | 表 A-62 |
| AUE_SOCKETSEND | 表 A-109 |
| AUE_STAT | 表 A-144 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|----------------|---------|
| AUE_STATFS | 表 A-145 |
| AUE_STATVFS | 表 A-146 |
| AUE_STIME | 表 A-147 |
| AUE_SYMLINK | 表 A-148 |
| AUE_SYSINFO | 表 A-149 |
| AUE_SYSTEMBOOT | 表 A-150 |
| AUE_UMOUNT | 表 A-151 |
| AUE_UNLINK | 表 A-152 |
| AUE_UTIME | 表 A-153 |
| AUE_UTIMES | 表 A-154 |
| AUE_UTSSYS | 表 A-155 |
| AUE_VFORK | 表 A-156 |
| AUE_VTRACE | 表 A-157 |

表 A-192 イベントからシステムコールへの変換 続く

| 監査イベント | システムコール |
|------------|---------|
| AUE_XMKNOD | 表 A-158 |
| AUE_XSTAT | 表 A-159 |

表 A-193 イベントからコマンドへの変換

| 監査イベント | コマンド |
|---------------------|---------|
| AUE_allocate_succ | 表 A-160 |
| AUE_allocate_fail | 表 A-161 |
| AUE_deallocate_succ | 表 A-162 |
| AUE_deallocate_fail | 表 A-163 |
| AUE_listdevice_succ | 表 A-164 |
| AUE_listdevice_fail | 表 A-165 |
| AUE_at_create | 表 A-166 |
| AUE_at_delete | 表 A-167 |
| AUE_at_perm | 表 A-168 |
| AUE_crontab_create | 表 A-169 |

表 A-193 イベントからコマンドへの変換 続く

| 監査イベント | コマンド |
|--------------------|---------|
| AUE_crontab_delete | 表 A-170 |
| AUE_cron_invoke | 表 A-171 |
| AUE_crontab_perm | 表 A-172 |
| AUE_halt_solaris | 表 A-173 |
| AUE_inetd_connect | 表 A-174 |
| AUE_init_solaris | 表 A-175 |
| AUE_ftpd | 表 A-176 |
| AUE_login | 表 A-177 |
| AUE_rlogin | 表 A-178 |
| AUE_telnet | 表 A-179 |
| AUE_logout | 表 A-180 |
| AUE_mountd_mount | 表 A-181 |
| AUE_mountd_umount | 表 A-182 |
| AUE_passwd | 表 A-183 |

表 A-193 イベントからコマンドへの変換 続く

| 監査イベント | コマンド |
|----------------------|---------|
| AUE_poweroff_solaris | 表 A-184 |
| AUE_reboot_solaris | 表 A-185 |
| AUE_rexd | 表 A-186 |
| AUE_rexecd | 表 A-187 |
| AUE_rshd | 表 A-188 |
| AUE_shutdown_solaris | 表 A-189 |
| AUE_su | 表 A-190 |
| AUE_uadmin_solaris | 表 A-191 |

BSM リファレンス

BSM は Solaris の運用環境に多数のユーティリティを提供します。この付録には、各ユーティリティが 4 つのセクションに分類され、それぞれが 1 つのリストに掲載されています。各表は、ユーティリティ名と各ユーティリティによって実行される処理の簡潔な説明を示します。各セクションは、マニュアルページの接尾辞で識別されています。

表 B-1 セクション 1M - 管理コマンド

| コマンド | 処理 |
|-------------------|--------------------------------|
| allocate(1M) | デバイスを割り当てる。 |
| audit(1M) | 監査デーモンを制御する。 |
| audit_startup(1M) | 監査サブシステムを初期化する。 |
| audit_warn(1M) | 監査デーモン警告スクリプトを実行する。 |
| auditconfig(1M) | 監査機能を構成する。 |
| auditd(1M) | 監査トレールファイルを制御する。 |
| auditreduce(1M) | 監査トレールファイルから監査レコードを組み合わせて選択する。 |
| auditstat(1M) | カーネル監査統計を表示する。 |

表 B-1 セクション 1M - 管理コマンド 続く

| コマンド | 処理 |
|------------------|---|
| bsmconv(1M) | Solaris システムで基本セキュリティモジュールを使用可能にする。 |
| bsmunconv(1M) | 基本セキュリティモジュールを使用不可にして、Solaris 運用環境に戻る (bsmconv(1M) のマニュアルページを参照)。 |
| deallocate(1M) | デバイスの割り当てを解除する。 |
| dminfo(1M) | デバイスマップファイル内のデバイスエントリに関する情報を表示する。 |
| list_devices(1M) | 割り当て可能デバイスをリストする。 |
| praudit(1M) | 監査トレールファイルの内容を出力する。 |

表 B-2 セクション 2 - システムコール

| システムコール | 処理 |
|-------------|----------------------------------|
| audit(2) | レコードを監査ログに書き込む。 |
| auditon(2) | 監査機能を操作する。 |
| auditsvc(2) | 監査ログを指定されたファイル記述子に書き込む。 |
| getaudit(2) | プロセス監査情報を取得する。 |
| getaudit(2) | ユーザ監査識別を取得する。 |
| setaudit(2) | プロセス監査情報を取得する (getaudit(2) を参照)。 |
| setaudit(2) | ユーザ監査識別を取得する (getaudit(2) を参照)。 |

表 B-3 セクション 3 - C ライブラリ関数

| ライブラリコール | 処理 |
|--|---|
| au_open(3), au_close(3), au_write(3) | 監査レコードを作成して書き込む。 |
| au_preselect(3) | 監査イベントを事前選択する。 |
| au_to_arg(3), au_to_attr(3), au_to_data(3), au_to_groups(3), au_to_in_addr(3), au_to_ipc(3), au_to_ipc_perm(3), au_to_iport(3), au_to_me(3), au_to_opaque(3), au_to_path(3), au_to_process(3), au_to_return(3), au_to_socket(3), au_to_text(3) | 監査レコードトークンを作成する (左のすべての関数については au_to(3) を参照)。 |
| au_user_mask(3) | ユーザのバイナリ事前選択マスクを 取得する。 |
| getacinfo(3), getacdir(3), getacflg(3), getacmin(3), getacna(3), setac(3), endac(3) | 監査制御ファイル情報を取得する。 |
| getauclassent(3), getauclassnam(3), setauclass(3), endauclass(3), getauclassnam_r(3), getauclassent_r(3) | audit_class エントリを取得す る。 |
| getauditflags(3), getauditflagsbin(3), getauditflagschar(3) | 監査フラグの仕様を変換する。 |
| getauevent(3), getauevnam(3), getauevnum(3), getauevnonam(3), setauevent(3), endauevent(3), getauevent_r(3), getauevnam_r(3), getauevnum_r(3) | audit_user エントリを取得する。 |
| getauusername(3), getauuserent(3), setauuser(3), endauuser(3) | audit_user エントリを取得する。 |
| getfauditflags(3) | プロセス監査状態を生成する。 |

表 B-4 セクション 4 - ヘッダ、テーブル、マクロ

| ファイル | 処理 |
|--------------------|---------------------------|
| audit.log(4) | 監査トレールファイルの形式を示す。 |
| audit_class(4) | 監査クラスの定義を示す。 |
| audit_control(4) | システム監査デーモン用の情報を制御する。 |
| audit_data(4) | 監査デーモンに関する現在の情報を保持する。 |
| audit_event(4) | 監査イベントの定義とクラスのマッピングを保持する。 |
| audit_user(4) | ユーザごとの監査データファイルを保持する。 |
| device_allocate(4) | 物理デバイス情報が入っている。 |
| device_maps(4) | 物理デバイス情報が入っている。 |

索引

数字

2 進形式の監査レコード形式, 45

A

accept 監査レコード, 96

access 監査レコード, 97

acct 監査レコード, 98

acl 監査レコード, 98

acl トークン, 80

adjtime 監査レコード, 99

administrative 監査クラス, 9

ad 監査フラグ, 9

all

監査クラス, 10

監査フラグ

使用上の注意, 11

説明, 10

ユーザ監査フィールド, 14

allhard 文字列、audit_warn スクリプト, 20

allocate 監査レコード

allocate-list device failure, 184

allocate-list device success, 183

deallocate device, 182

deallocate device failure, 183

device allocate failure, 182

device allocate success, 182

allocate コマンド

オプション, 63

使用, 76

割り当て機構の機能, 72, 74

allsoft 文字列、audit_warn スクリプト, 19

always-audit フラグ

説明, 14

プロセス事前選択マスク, 15

application 監査クラス, 10

ap 監査フラグ, 10

arbitrary トークン, 48, 49, 80

Archive テープドライブクリーンスクリプト, 67

arge 方針

exec_env トークン, 83

フラグ, 41

argv 方針

exec_args トークン, 83

arg トークン, 49, 81

attr トークン, 49, 82

at 監査レコード

at-create crontab, 184

at-delete atjob, 185

at-permission, 185

AUDIOGETREG ioctl システムコール, 70

AUDIOSETREG ioctl システムコール, 70

audio_clean スクリプト, 70

AUDIO_DRAIN ioctl システムコール, 70

AUDIO_SETINFO ioctl システムコール, 70

audit -n コマンド, 18

audit -s コマンド

監査ファイルの再読み込み, 17

既存のプロセスの事前選択マスク, 13

ディレクトリポインタのリセット, 18

audit -t コマンド, 16

auditconfig コマンド

オプション, 38, 40

接頭辞、フラグ, 11

- 引数としての監査フラグ, 9
- 必要な記憶装置容量の縮小, 25
- フラグ接頭辞, 12
- auditconfig コマンドの -setsmask オプション, 40
- auditconfig コマンドの -chkconf オプション, 39
- auditconfig コマンドの -conf オプション, 39
- auditconfig コマンドの -getclass オプション, 39
- auditconfig コマンドの -getcond オプション, 39
- auditconfig コマンドの -getpinfo オプション, 40
- auditconfig コマンドの -getpolicy オプション, 40
- auditconfig コマンドの -lsevent オプション, 39
- auditconfig コマンドの -lspolicy オプション, 40
- auditconfig コマンドの -setclass オプション, 39
- auditconfig コマンドの -setcond オプション, 39
- auditconfig コマンドの -setpmask オプション, 40
- auditconfig コマンドの -setpolicy オプション, 40
- auditconfig コマンドの -setumask オプション, 40
- auditon 監査レコード
 - A_GETCAR コマンド, 100
 - A_GETCLASS コマンド, 100
 - A_GETCOND コマンド, 101
 - A_GETCWD コマンド, 101
 - A_GETKMASK コマンド, 102
 - A_GETSTAT コマンド, 102
 - A_GPOLICY コマンド, 103
 - A_GQCTRL コマンド, 103
 - A_SETCLASS コマンド, 104
 - A_SETCOND コマンド, 104
 - A_SETKMASK コマンド, 105
 - A_SETSMASK コマンド, 105
 - A_SETSTAT コマンド, 106
 - A_SETUMASK コマンド, 106
 - A_SPOLICY コマンド, 107
 - A_SQCTRL コマンド, 107
- auditreduce コマンド, 20, 23
 - a オプション, 58
 - b オプション, 58
 - d オプション, 57
 - m オプション, 58
 - O オプション, 26, 30, 57
 - not_terminated ファイルの整理, 30, 57
 - オプション, 21, 58
 - オプションを指定しない, 21, 23
 - 機能, 55
 - 説明, 20, 21, 45, 55
 - タイムスタンプの使用, 29
 - 分散システム, 56
 - 例, 56, 57
- auditreduce コマンドの -a オプション, 58
- auditreduce コマンドの -b オプション, 58
- auditreduce コマンドの -m オプション, 58
- auditreduce コマンドの -O オプション, 26, 57
- auditsvc
 - 監査レコード, 108
 - システムコール
 - 失敗, 20, 95
- audit_control ファイル
 - audit_user ファイルの改良, 14
 - dir: 行
 - files サブディレクトリ, 32
 - 説明, 12
 - 例, 13, 33
 - flags: 行
 - 接頭辞, 11, 12
 - 説明, 12
 - プロセス事前選択マスク, 15
 - flags: 行接頭辞, 11, 12
 - minfree: 行
 - audit_warn 条件, 19
 - naflags, 12
 - 概要, 12, 13
 - 終了後の監査データの再読み込み, 13
 - 内容のエラー, 20
 - 例, 13, 33
- audit_control ファイルの dir: 行
 - files サブディレクトリ, 32
 - 説明, 12
 - 例, 13, 33
- audit_control ファイルの flags: 行
 - 接頭辞, 11, 12
- audit_control ファイルの minfree: 行

- audit_warn 条件, 19
- 空き領域の決定, 35
- 説明, 12
- audit_control ファイルの行
 - プロセス事前選択マスク, 15
- audit_control ファイルの非帰因フラグ, 12
- audit_data ファイル, 17
- audit_user ファイル
 - フラグ接頭辞, 11, 12
 - プロセス事前選択マスク, 15
 - ユーザ監査フィールド, 14
- audit_warn スクリプト, 19, 20
- allhard 文字列, 20
- allsoft 文字列, 19
- auditsvc 文字列, 20
- ebusy 文字列, 20
- hard 文字列, 19
- postsigterm 文字列, 20
- soft 文字列, 19
- tmpfile 文字列, 20
- 監査デーモンの実行, 17
- 条件の検出, 19, 20
- 説明, 17, 19
- audit 監査レコード, 99
- audit トークン
 - acl トークン, 80
- audit フラグ
 - auditconfig コマンドオプション, 40
- audit レコード
 - 方針フラグ, 40
 - ユーザが読める書式に変換, 21
- audit レコードをユーザが読める書式
 - 監査レコードを変換, 59
- AUE... 名, 7
- event-to-システムコールへの変換テーブル, 197, 209

B

- bind 監査レコード, 108
- bsmconv スクリプト
 - BSM を使用可能にする, 2
 - device_maps ファイルの作成, 65
- bsmunconv スクリプト, 2
- BSM (基本セキュリティモジュール)
 - インストール, 2, 3
 - クライアントとサーバの関係, 2
 - 使用可能にする, 2

- 使用不可能にする, 2
- パッケージ, 2
- BSM のインストール, 2
- BSM を使用不可能にする, 2

C

- C2 TCSEC の機能, 62
- CD-ROM ドライブ, ix
 - デバイスクリーンスクリプト, 70
- chdir 監査レコード, 109
- chmod 監査レコード, 109
- chown 監査レコード, 110
- chroot 監査レコード, 110
- close 監査レコード, 111
- cl 監査フラグ, 9
- cnt 方針, 36, 37
 - flag, 41
- connect 監査レコード, 111
- creat 監査レコード, 112
- crontab 監査レコード
 - cron-invoke atjob or crontab, 187
 - crontab-crontab created, 186
 - crontab-crontab deleted, 186
 - crontab-permission, 187
- cron ジョブ, 18

D

- deallocate コマンド
 - 使用, 76
 - 説明, 64, 75
 - デバイスクリーンスクリプト, 71
 - 割り当てエラー状態, 64
- device_allocate ファイル, 67, 68
 - 概要, 66, 69
 - フォーマット, 67, 68
- device_maps ファイル
 - 概要, 65
 - フォーマット, 65, 66
- dminfo コマンド, 65
- doorfs 監査レコード
 - DOOR_BIND コマンド, 113
 - DOOR_CALL コマンド, 113
 - DOOR_CREATE コマンド, 114
 - DOOR_CRED コマンド, 114
 - DOOR_INFO コマンド, 115

DOOR_RETURN コマンド, 115
DOOR_REVOKE コマンド, 116
DOOR_UNBIND コマンド, 116
-d オプション
auditreduce コマンド, 57
praudit コマンド, 59

E

ebusy 文字列、 audit_warn スクリプト, 20
eject コマンド, 70
enter prom 監査レコード, 117
/etc/security/audit/bsmconv スクリプト
BSM を使用可能にする, 2
device_maps ファイルの作成, 65
/etc/security/audit/bsmunconv スクリプト, 2
/etc/security/audit_data ファイル, 17
/etc/security/audit_event ファイル, ix
概要, 7, 8
監査イベントのタイプ, 45
/etc/security/audit_startup ファイル, 6
/etc/security/audit_warn スクリプト, 17, 19, 20
/etc/security/audit ディレクトリ, 27, 31
/etc/security/dev ロックファイル, 71, 74
/etc/security ディレクトリ, 31
execve 監査レコード, 118
exec_args トークン, 83
exec_env トークン, 83
exec 監査クラス, 10
exec 監査レコード, 117
exit prom 監査レコード, 118
exit 監査レコード, 119
exit トークン, 50, 83
ex 監査フラグ, 10

F

facl 監査レコード, 119
fa 監査フラグ, 9
fchdir 監査レコード, 120
fchmod 監査レコード, 121
fchown 監査レコード, 121
fchroot 監査レコード, 122
fcntl 監査レコード, 123
fc 監査フラグ, 9
fd_clean スクリプト, 70

fd 監査フラグ, 9
files サブディレクトリ, 32
file vnode トークン, 49
file_attr_acc 監査クラス, 9
file_attr_mod 監査クラス, 9
file_close 監査クラス, 9
file_creation 監査クラス, 9
file_deletion 監査クラス, 9
file_read 監査クラス, 9
file_write 監査クラス, 9
file トークン, 50, 84
flags: 行 in audit_control ファイル
説明, 12
fm 監査フラグ, 9
fork1 監査レコード, 124
fork 監査レコード, 123
fr 監査フラグ, 9
fstafs 監査レコード, 124
ftpd login 監査レコード, 189
fw 監査フラグ, 9
-F オプション
allocate コマンド, 63
deallocate コマンド, 64
st_clean スクリプト, 71

G

getaudit 監査レコード, 125
getaudit 監査レコード, 125
getmsg 監査レコード, 126
socket accept, 126
socket receive, 127
getpmsg 監査レコード, 127
getportaudit 監査レコード, 128
groups トークン, 50, 84, 85
group 方針
flag, 41
groups トークン, 50, 84, 85
newgroups トークン, 89

H

halt: machine halt 監査レコード, 188
hard 文字列、 audit_warn スクリプト, 19
header トークン
praudit 表示, 48
イベント修飾子フィールドフラグ, 86

監査レコードの順序, 46, 85
記述, 85, 86
説明, 47
フィールド, 47
フォーマット, 85

I

ID

auditconfig コマンドオプション, 40
監査, 6, 16, 44
監査セッション, 16, 44
監査ユーザ, 44
端末, 16
in.ftpd 監査レコード, 189
in.rexecd 監査レコード, 195
in.rshd: rshd アクセス拒否/許可監査レコード, 195
inetd: inetd サービス要求の監査レコード, 188
init: init サービス要求の監査レコード, 189
inst_sync 監査レコード, 128
in_addr トークン, 50, 86
ioctl: 特定のデバイスに対する ioctl の監査レコード, 129
ioctl 監査クラス, 10
ioctl システムコール, 10, 70
io 監査フラグ, 10
ipc_perm トークン, 51, 88
ipc 監査クラス, 9
ipc タイプフィールド (ipc トークン), 87
ipc トークン, 51, 87
ipport トークン, 52, 88
ip 監査フラグ, 9
ip トークン, 51, 87
-I オプション
 deallocate コマンド, 64
 st_clean スクリプト, 71

K

kill 監査レコード, 131

L

lchown 監査レコード, 131
link 監査レコード, 132
list_devices コマンド, 64, 76
login_logout 監査クラス, 9

login 監査レコード
 logout, 191
 rlogin, 190
 telnet login, 191
 terminal login, 190
lo 監査フラグ, 9
lstat 監査レコード, 132
lxstat 監査レコード, 133
-l オプション
 praudit コマンド, 59

M

machine halt 監査レコード, 188
machine reboot 監査レコード, 194
memcntl 監査レコード, 133
mkdir 監査レコード, 134
mknod 監査レコード, 134
mmap 監査レコード, 135
modctl 監査レコード
 MODADDMAXBIND コマンド, 136
 MODCONFIG コマンド, 137
 MODLOAD コマンド, 137
 MODUNLOAD コマンド, 138
mountd 監査レコード
 NFS mount 要求, 192
 NFS unmount 要求, 192
mount 監査レコード, 138
msgctl 監査レコード
 IPC_RMID コマンド, 139
 IPC_SET コマンド, 140
 IPC_STAT コマンド, 140
msgget 監査レコード, 141
msgrcv 監査レコード, 141
msgsnd 監査レコード, 142
mt コマンド、デバイスのクリーンアップオプション, 69
mt コマンドの rewoffl オプション, 69
munmap 監査レコード, 142

N

naflags : audit_control ファイルの行, 12
na 監査フラグ, 9
network 監査クラス, 9
never-audit フラグ, 14
newgroups トークン, 89

NFS mount 要求の監査レコード, 192
NFS unmount 要求の監査レコード, 192
nice 監査レコード, 143
non_attrib 監査クラス, 9
not_terminated ファイルの整理, 30, 57
no_class 監査クラス, 9
no 監査フラグ, 9
nt 監査フラグ, 9
null 監査クラス, 9

O

opaque トークン, 52, 90
open 監査レコード
 read, 143
 read, create, 144
 read, create, truncate, 144
 read, truncate, 145
 read, write, 145
 read, write, create, 146
 read, write, create, truncate, 146
 read, write, truncate, 147
 write, 147
 write, create, 148
 write, create, truncate, 148
 write, truncate, 149
other 監査クラス, 10
ot 監査フラグ, 10
-O オプション, 30

P

passwd 監査レコード, 193
pathconf 監査レコード, 150
path トークン, 52, 90
path 方針フラグ, 41
pc 監査フラグ, 9
pipe 監査レコード, 150
postsigterm 文字列、audit_warn スクリプト, 20
poweroff 監査レコード, 193
praudit コマンド, ix
 auditreduce 出力をパイプする, 56
 監査レコードをユーザが読める書式に変換する, 21
 出力書式, 58, 59
 使用, 58, 59
 説明, 45

ユーザが読める書式, 46, 55
ユーザが読める書式に監査レコードを変換, 8

priocntlsys 監査レコード, 151
process dumped core 監査レコード, 151
process groups トークン
 groups トークン, 50
processor_bind 監査レコード, 152
process 監査クラス, 9
process トークン, 53, 91
putmsg 監査レコード, 153
 socket connect, 154
 socket send, 154
putpmsg 監査レコード, 155
p_online 監査レコード, 149

R

raw praudit 出力書式, 59
 header トークン, 48
readlink 監査レコード, 155
reboot: マシンリブート監査レコード, 194
rename 監査レコード, 156
return トークン, 53, 91
rmdir 監査レコード, 156
rpc.rexd 監査レコード, 194
-r praudit 出力書式, 59
 header トークン, 48
rshd アクセス拒否/許可監査レコード, 195

S

/sbin/init 監査レコード, 189
SCSI デバイス, ix
 st_clean スクリプト, 67
secondary 監査ディレクトリ, 13
semctl 監査レコード
 GETALL コマンド, 157
 GETNCNT コマンド, 157
 GETPID コマンド, 158
 GETVAL コマンド, 158
 GETZCNT コマンド, 159
 IPC_RMID コマンド, 159
 IPC_SET コマンド, 160
 IPC_STAT コマンド, 161
 SETALL コマンド, 160
 SETVAL コマンド, 161

- semget 監査レコード, 162
- semop 監査レコード, 162
- seq トークン, 53, 92
- seq 方針フラグ, 41
- setaudit 監査レコード, 163
- setaudit 監査レコード, 164
- setegid 監査レコード, 165
- seteuid 監査レコード, 165
- setgid 監査レコード, 165
- setgroups 監査レコード, 166
- setpgrp 監査レコード, 166
- setregid 監査レコード, 167
- setreuid 監査レコード, 167
- setrlimit 監査レコード, 168
- setuid 監査レコード, 168
- shmat 監査レコード, 169
- shmctl 監査レコード
 - IPC_RMID コマンド, 169
 - IPC_SET コマンド, 170
 - IPC_STAT コマンド, 170
- shmdt 監査レコード, 171
- shmget 監査レコード, 171
- shutdown 監査レコード, 172, 196
- socket-inet トークン, 93
- socket accept 監査レコード, 126
- socket connect 監査レコード, 154
- socket receive 監査レコード, 127
- socket send 監査レコード, 154
- socket トークン, 54, 92
- soft 文字列、audit_warn スクリプト, 19
- s praudit 出力書式, 59
 - header トークン, 48
- sr_clean スクリプト, 70
- statfs 監査レコード, 174
- statvfs 監査レコード, 174
- stat 監査レコード, 173
- stime 監査レコード, 175
- st_clean スクリプト for テープドライブ, 69
- st_clean スクリプトの -S オプション, 71
- subject トークン, 54, 93
- success
 - turning off 監査フラグ for, 11
- SUNWcar パッケージ, 2
- SUNWcsr パッケージ, 2
- SUNWcsu パッケージ, 2
- SUNWhea パッケージ, 2
- SUNWman パッケージ, 2

- su 監査レコード, 196
- symlink 監査レコード, 175
- sysinfo 監査レコード, 176
- system booted 監査レコード, 176
- System V IPC
 - ipc_perm トークン, 51, 88
 - ipc 監査クラス, 9
 - ipc トークン, 51, 87

T

- TCP アドレス, 52, 88
- TCSEC (Trusted Computer System Evaluation Criteria) C2 の機能, 62
- text トークン, 55, 94
- tmpfile 文字列、audit_warn スクリプト, 20
- trailer トークン
 - praudit 表示, 48
 - 監査レコードの順序, 46, 95
 - 記述, 95
 - 形式, 95
 - 説明, 48
 - フィールド, 48
- trail 方針フラグ, 41

U

- uadmin 監査レコード, 197
- UDP アドレス, 52, 88
- umount: old version 監査レコード, 177
- unlink 監査レコード, 177
- /usr/bin/at 監査レコード
 - at-create crontab, 184
 - at-delete atjob, 185
 - at-permission, 185
- /usr/bin/crontab 監査レコード
 - cron-invoke atjob or crontab, 187
 - crontab-crontab created, 186
 - crontab-crontab deleted, 186
 - crontab-permission, 187
- /usr/bin/login 監査レコード
 - logout, 191
 - rlogin, 190
 - telnet login, 191
 - terminal login, 190
- /usr/bin/passwd: passwd 監査レコード, 193
- /usr/bin/su 監査レコード, 196

/usr/lib/nfs/mountd 監査レコード
NFS mount 要求, 192
NFS unmount 要求, 192
/usr/sbin/allocate 監査レコード
allocate-list device failure, 184
allocate-list device success, 183
deallocate device, 182
deallocate device failure, 183
device allocate failure, 182
device allocate success, 182
/usr/sbin/halt 監査レコード, 188
/usr/sbin/in.ftpd 監査レコード, 189
/usr/sbin/in.rexecd 監査レコード, 195
/usr/sbin/in.rshd 監査レコード, 195
/usr/sbin/inetd 監査レコード, 188
/usr/sbin/init 監査レコード, 189
/usr/sbin/poweroff 監査レコード, 193
/usr/sbin/reboot 監査レコード, 194
/usr/sbin/rpc.rexd 監査レコード, 194
/usr/sbin/shutdown 監査レコード, 189
/usr/sbin/uadmin 監査レコード, 197
/usr/ucb/shutdown 監査レコード, 196
utimes 監査レコード, 178
utime 監査レコード, 178
utssys - fusers 監査レコード, 179
-U オプション
allocate コマンド, 64
list_devices コマンド, 64

V

vfork 監査レコード, 179
vnode トークン, 49, 82
vtrace 監査レコード, 180

X

xmknod 監査レコード, 180
xstat 監査レコード, 181
Xylogics テープドライブクリーンアップ
ト, 67

記号

+ 監査フラグ接頭辞, 10, 12, 36
- 監査フラグ接頭辞, 10, 12, 36

あ

アスタリスク (*) device_allocate ファイル, 67,
68
新しいデバイスクリーンスクリプトを書く, 71

い

一次監査ディレクトリ, 12, 31
一次ファイルを使用できない場合, 20
イベント, ix
event-to-システムコールへの変換テーブ
ル, 197, 209

概要, 7, 8

カテゴリ, 7

監査トルールに含まれる, 7

クラスのマッピング, 7, 41

番号, 7

レコード形式と, 45

カーネルイベント

auditconfig コマンドオプション, 39

監査トークン, 46

説明, 7

ユーザレベルのイベント

auditconfig コマンドオプション, 39

監査トークン, 46

説明, 7

イベント修飾子フィールド (header トーク
ン), 86

イベント番号, 7

インストール BSM, 3

インターネット関連トークン

in_addr トークン, 50, 86

ipport トークン, 52, 88

ip トークン, 51, 87

socket-inet トークン, 93

socket トークン, 54, 92

え

エクスポートリスト, 28

エラー

監査ディレクトリがいっぱいになる, 17,
20, 95

内部エラー, 20

割り当てエラー状態, 64

お

- オーディオドライブ
 - デバイスクリーンスクリプト, 70
- オブジェクトの再利用の要件, 62, 69, 70
 - デバイスクリーンスクリプト
 - CD-ROM ドライブ, 70
 - 新しいスクリプトの作成, 71
 - 新しいスクリプトを書く, 71
 - オーディオドライブ, 70
 - 説明, 69
 - テープドライブ, 67, 69
 - デバイスの追加, 75
 - フロッピーディスクドライブ, 70

か

- カーネルイベント, ix
 - auditconfig コマンドオプション, 39, 40
 - 監査トークン, 46
 - 監査レコード, 96, 181
 - 説明, 7
- 格納コスト, 24, 25
- 監査 ID, 6, 16, 44
- 監査イベント, ix
 - event-to-システムコールへの変換テーブル, 197, 209
 - 概要, 7, 8
 - カテゴリ, 7
 - 監査トレールに含まれる, 7
 - クラスのマッピング, 7, 41
 - 番号, 7
 - レコード形式と, 45
 - カーネルイベント
 - auditconfig コマンドオプション, 39
 - 監査トークン, 46
 - 説明, 7
 - 監査イベントファイル
 - 概要, 7, 8
 - 監査イベントのタイプ, 45
 - ユーザーレベルのイベント
 - auditconfig コマンドオプション, 39
 - 監査トークン, 46
 - 説明, 7
- 監査イベントファイル, ix
 - 概要, 7, 8
 - 監査イベントのタイプ, 45

監査管理

- audit_user ファイル監査フィールド, 14
- 監査トレールの作成, 18
- コスト制御, 23
- 通常のユーザの監査, 25
- auditreduce コマンド
 - O オプション, 30
- 監査イベント
 - event-to-システムコールへの変換テーブル, 197, 209
- 監査トレールの作成
 - 適切なディレクトリ, 18
- 監査ファイル
 - file トークン, 84
 - 概要, 27
- 監査フラグ
 - audit_user ファイル, 14
- 構成
 - 監査トレールのオーバーフローの防止, 37
 - プロセス監査特性
 - 監査 ID, 16
- 監査管理アカウント, 36, 37
- 監査起動ファイル, 6
- 監査機能の管理
 - audit_user ファイル監査フィールド, 14
 - audit_warn スクリプト, 17, 19, 20
 - auditreduce コマンド, 20, 23, 55, 58
 - 概要, 6
 - 監査管理アカウント, 36, 37
 - 監査トレールのオーバーフローの防止, 37, 38
 - 監査トレールの作成, 16
 - 監査パーティション, 31, 33
 - 監査ファイル, 27, 30
 - 監査フラグ, 8, 12
 - 監査レコード, 8
 - 起動, 6
 - 効率, 25, 27
 - コスト制御, 25
 - プロセスの監査特性, 15, 16
 - audit_control ファイル
 - audit_user ファイルの改良, 14
 - flags: 行接頭辞, 11, 12
 - 概要, 12, 13
 - 内容のエラー, 20

- auditreduce コマンド
 - 説明, 21
 - a オプション, 58
 - b オプション, 58
 - d オプション, 57
 - O オプション, 26, 57
 - not_terminated ファイルの整理, 30
 - not_terminated ファイルの変更, 57
 - オプション, 21, 58
 - オプションを指定しない, 21, 23
 - 機能, 55
 - 説明, 20, 45, 55
 - タイムスタンプの使用, 29
 - 分散システム, 56
 - 例, 56, 57
- audit ファイル
 - 許可, 32
- audit フラグ
 - auditconfig コマンドオプション, 40
- 監査イベント
 - auditconfig コマンドオプション, 39
 - カーネルイベント, 7, 39, 46
 - 概要, 7, 8
 - カテゴリ, 7
 - 監査トークン, 46
 - 監査トレールに含まれる, 7
 - クラスのマッピング, 7, 41
 - 番号, 7
 - ユーザレベルのイベント, 7, 39, 46
 - レコード形式と, 45
- 監査クラス
 - auditconfig コマンドオプション, 39
 - イベントのマッピング, 7, 41
 - 概要, 7, 8
 - 監査のための選択, 7
 - 定義の変更, 42
 - フラグと定義, 9, 10
- 監査トレールの作成
 - audit_data ファイル, 17
 - 概要, 16
 - 監査デーモンの役割, 17, 18
 - 監査ファイルのサイズの管理, 18
- 監査ファイル
 - auditreduce コマンド, 20, 23
 - file トークン, 50
 - not_terminated マークが付いたアク
タイプでないファイル, 30
 - 新しいファイルに切り替える, 18
 - 印刷, 56
 - 概要, 28
 - 組み合わせる, 20, 23, 26
 - サイズの管理, 18
 - 縮小, 20, 23, 26
 - 使用されないファイルの
not_terminated のマーク, 57
 - 全体を表示する, 56
 - 操作の順序, 12
 - タイムスタンプ, 29
 - ディレクトリの位置, 27, 31, 32
 - 必要な記憶装置容量の縮小, 24 - 26
 - ファイルシステムの最小空き領域, 12
 - ユーザ名, 28, 30
 - ログイン/ログアウトメッセージを 1
つのファイルにコピーす
る, 57
- 監査フラグ
 - audit_control ファイル行, 12
 - audit_user ファイル, 14
 - 概要, 8
 - 構文, 10, 11
 - 接頭辞, 11, 12
 - 定義, 9, 10
 - プロセス事前選択マスク, 15
 - 方針フラグ, 40
 - マシン全体, 8, 12
- 構成
 - auditconfig コマンド, 38, 40
 - 概要, 33, 34
 - 監査トレールのオーバーフローの防
止, 38
 - 監査方針の設定, 40
 - 計画, 34, 37
- コスト制御
 - 記憶装置, 24, 25
 - 処理時間, 23
 - 分析, 23

- プロセス監査特性
 - 監査セッション ID, 16
 - 端末 ID, 16
 - プロセス事前選択マスク, 24, 25
- プロセスの監査特性
 - プロセス事前選択マスク, 15
- 監査クラス
 - auditconfig コマンドオプション, 39
 - イベントのマッピング, 7, 41
 - 概要, 7, 8
 - 監査のための選択, 7
 - 定義の変更, 42
 - フラグと定義, 9, 10
- 監査サーバのマウントポイントのパス名, 31
- 監査しきい値, 12
- 監査セッション ID, 16, 44
- 監査ディレクトリのマウント, 28
- 監査データをリアルタイムで監視する, 26
- 監査デーモン
 - audit_control ファイルの再読み込み, 13
 - audit_startup ファイル, 6
 - 監査起動ファイル, 6
 - 監査トレールの作成, 16 - 18
 - 監査を使用可能にする, 6
 - 機能, 17
 - 終了, 16
 - 適切なディレクトリ, 18
 - 開かれた監査ファイルの順序, 12
 - audit_warn スクリプト
 - 実行, 17
 - 条件の検出, 19, 20
 - 説明, 17, 19
- 監査デバイス, ix
- 監査トークン, ix
 - arbitrary トークン, 48, 49, 80
 - arg トークン, 49, 81
 - attr トークン, 49, 82
 - exec_args トークン, 83
 - exec_env トークン, 83
 - exit トークン, 50, 83
 - file トークン, 50, 84
 - groups トークン, 50, 84, 85
 - header トークン, 46 - 48, 85, 86
 - in_addr トークン, 50, 86
 - ipc_perm トークン, 51, 88
 - ipc トークン, 51, 87
 - ipport トークン, 52, 88
 - ip トークン, 51, 87
 - newgroups トークン, 89
 - opaque トークン, 52, 90
 - path トークン, 52, 90
 - process トークン, 53, 91
 - return トークン, 53, 91
 - seq トークン, 53, 92
 - socket-inet トークン, 93
 - socket トークン, 54, 92
 - subject トークン, 54, 93
 - text トークン, 55, 94
 - trailer トークン, 46, 48, 95
- 監査レコードの順序, 46
- 監査レコードフォーマット, 45, 55, 77, 78
- 説明, 8
- タイプ, 46
- テーブル, 78, 79
- 方針フラグ, 40
- 監査トレール, ix
 - イベントを含む, 7
 - オーバーフローの防止, 37, 38
 - 作成, 16, 18, 27
 - すべてのファイルをマージする, 21, 23
 - ディレクトリの位置, 27, 31, 32
 - 分析, 43, 59
 - リアルタイムで監視する, 26
 - 作成
 - audit_data ファイル, 17
 - 概要, 16
 - 監査デーモンの役割, 16 - 18
 - 監査ファイルのサイズの管理, 18
 - 適切なディレクトリ, 18
- 分析
 - auditreduce コマンド, 45, 55, 58
 - praudit コマンド, 45, 58, 59
 - 監査機能, 43, 44
 - 監査レコード形式, 45, 55
 - コスト, 23
 - ツール, 44, 45
- 監査トレールのオーバーフローの防止, 37, 38
- 監査トレールの作成, 16, 18
- audit_data ファイル, 17
- 概要, 16
- 監査デーモンの役割, 17, 18
- 監査ファイルのサイズの管理, 18
- 適切なディレクトリ, 18

- 監査の終了時に受け取る信号, 20
- 監査パーティション, 31, 33
- 監査ファイル, ix, 27, 30
 - auditreduce コマンド, 20, 23
 - file トークン, 50, 84
 - not_terminated マークが付いたアクティブでないファイル, 30
 - 新しいファイルに切り替える, 18
 - 印刷, 56
 - 概要, 27, 28
 - 許可, 32
 - 組み合わせる, 20, 23, 26
 - サイズの管理, 18
 - 縮小, 20, 23, 26
 - 使用されないファイルの not_terminated のマーク, 57
 - 全体を表示する, 56
 - 操作の順序, 12
 - タイムスタンプ, 29
 - ディレクトリの位置, 27, 31, 32
 - 必要な記憶装置容量の縮小, 24 - 26
 - ファイルシステムの最小空き領域, 12
 - ユーザ名, 28, 30
 - ログイン/ログアウトメッセージを 1 つのファイルにコピーする, 57
 - ユーザ名
 - 使用, 29
 - 書式, 28
 - タイムスタンプ, 29
 - 閉じられたファイル, 29
 - まだアクティブなファイル, 29
- 監査ファイルシステム, 32
- 監査ファイル縮小, 26
 - auditreduce コマンド, 20, 23
 - 必要な記憶装置容量, 24 - 26
- 監査ファイルのタイムスタンプ, 29
- 監査ファイルを組み合わせる, 26
 - auditreduce コマンド, 20, 23
- 監査フラグ, 8, 12
 - audit_control ファイル行, 12
 - audit_user ファイル, 14
 - 概要, 8
 - 構文, 10, 11
 - 接頭辞, 10, 12, 36
 - 定義, 9, 10
 - プロセス事前選択マスク, 15
 - 方針フラグ, 40
 - マシン全体, 8, 12

- 監査フラグ接頭辞, 10 - 12, 36
- 監査方針, ix
 - auditconfig オプション, 40
 - 設定, 40
- 監査レコード, ix, 77, 197
 - カーネルレベルで生成される, 96, 181
 - 概要, 8
 - 監査ディレクトリがいっぱいになる, 17, 20, 95
 - 監査ファイル縮小, 26
 - 形式または構造, 45, 55, 77, 96
 - 選択, 45
 - それ自身で意味のわかるレコード, 44
 - ツール, 44, 45
 - 表示, 45
 - ユーザが読める書式に変換, 8, 45, 58, 59
 - ユーザレベルでの生成, 181, 197
- 監査レコードの選択, 45
- 監査レコードをユーザが読める書式に監査レコードを変換する, 58
- 説明, 55
- 監査レコードをユーザが読める書式に変換, 45, 58, 59
- 監査ログを印刷する, 56
- 管理監査, ix

き

- 記憶装置のオーバーフローの防止, 37, 38
- キャレット (^) 監査フラグ接頭辞, 11, 12
- 強制クリーンアップ, 71

く

- クライアント、BSM を使用可能にする, 3
- クラス
 - auditconfig コマンドオプション, 39
 - イベントのマッピング, 7, 41
 - 概要, 7, 8
 - 監査のための選択, 7
 - 定義の変更, 42
 - フラグと定義, 9, 10

こ

- 構成
 - auditconfig コマンド, 38, 40

概要, 33, 34
監査トレールのオーバーフローの防
止, 37, 38
監査方針の設定, 40
計画, 34, 37
項目サイズフィールド値 (arbitrary トーク
ン), 81
効率, 25, 27
コスト制御, 23, 25
記憶装置, 24, 25
処理時間, 23
分析, 23
コマンド, ix
デバイスの割り当てユーティリティ, 63,
64
コマンドの -a オプション, 58
コマンドの割り当て, ix
コメント
device_allocate ファイル, 67
device_maps ファイル, 65

さ
サーバ、クライアント上で BSM を使用可能に
する, 3
サイズ
監査ファイル縮小, 26
監査ファイルの管理, 18
監査ファイル縮小
auditreduce コマンド, 20, 23
必要な記憶装置容量, 24 - 26

し
システムコール
arg トークン, 49, 81
auditvc が失敗, 20, 95
close, 9
event-to-システムコールへの変換テーブ
ル, 197, 209
exec_args トークン, 83
exec_env トークン, 83
ioctl, 10, 70
return トークン, 53, 91
イベント番号, 7
事前選択マスク
auditconfig コマンドオプション, 40
記憶装置容量の縮小コスト, 24, 25

説明, 15
マシン全体, 12
失敗
監査フラグ接頭辞, 10, 11
監査フラグのオフ, 11
監査フラグを切る, 12
自動的に監査を使用可能にする, 6
終了
BSM を使用不可にする, 2
監査デーモン, 16
監査デーモンを終了させる, 16
監査の終了時に受け取る信号, 20
出力形式フィールド値 (arbitrary トークン), 80
使用可能にする
BSM, 2
監査, 6
処理時間のコスト, 23

せ
成功
監査フラグ接頭辞, 10, 11
セキュリティデバイスの使用に伴うリスク, 62
セッション ID, 16, 44

そ
ソフト限界
audit_warn 条件, 19
ソフト制限
minfree: 行の説明, 12
空き領域の決定, 35

た
端末 ID, 16

つ
通常の利用者の監査, 25

て
ディスクドライブ, ix
ディスクフルマシンの監査ディレクトリ, 28,
31
ディスク容量の要件, 24, 25

ディスクレスクライアント、BSM を使用可能にする, 3

ディレクトリ, ix

audit_control ファイル定義, 12

files サブディレクトリ, 32

監査ディレクトリがいっぱいになる, 17, 20, 95

監査ディレクトリの位置, 27, 31, 32

監査ディレクトリのマウント, 28

監査デーモンポインタ, 18

監査パーティション, 31, 33

許可, 32

ディスクフルマシンの, 28, 31

適切な監査デーモン, 18

テープデバイス, ix

テープドライブ

st_clean スクリプト, 67

使用に伴うリスク, 62

デバイスクリプト, 69

テープドライブ用 st_clean スクリプト, 67

デバイス, ix

管理, 74

追加, 74

ロックファイル, 71, 74

デバイスクリプト

CD-ROM ドライブ, 70

新しいスクリプトの作成, 71

オーディオドライブ, 70

オプション, 71

説明, 69

テープドライブ, 67, 69

デバイスの追加, 75

フロッピーディスクドライブ, 70

デバイスの管理, 74

デバイスの使用に伴うリスク, 62

デバイスの追加, 74

デバイスの割り当て, 62, 76

device_allocate ファイル, 66, 68

device_maps ファイル, 65, 66

list_devices コマンド, 64, 75

再度割り当てる, 63

デバイスクリプト, 69, 71

デバイスの使用に伴うリスク, 62

デバイスの追加, 74

デバイスの割り当て, 76

デバイスの割り当ての使用, 76

デバイスを管理する, 74

ユーティリティ, 63, 64

ロックファイルの設定, 71, 74

割り当てエラー状態, 64

割り当て可能デバイス, 67, 74

割り当て機構の構成要素, 62

allocate コマンド

オプション, 63

使用, 76

割り当て機構の機能, 72, 74

deallocate コマンド

使用, 76

説明, 64, 75

デバイスクリプトと, 71

割り当てエラー状態, 64

デバイスクリプト

CD-ROM ドライブ, 70

新しいスクリプトの作成, 71

オーディオドライブ, 70

オプション, 71

説明, 69

テープドライブ, 67, 69

デバイスの追加, 75

フロッピーディスクドライブ, 70

デバイスを再度割り当てる, 63

デバッグシーケンス番号, 53, 92

デフォルト

praudit 出力書式, 59

監査起動ファイル, 6

監査方針, 40

マシン全体, 8

praudit 出力書式

header トークン, 48

な

名前

デバイス名

device_allocate ファイル, 68

device_maps ファイル, 65

に

二次監査ディレクトリ, 31

日時 auditreduce コマンドオプション, 58

は

パーティション、監査, 31, 33

ハードディスク容量の要件, 24, 25
バックスラッシュ (\) ファイル行の終了記号, 65, 67
番号、イベント, 7

ひ

表記上の規則, x
表示
 監査レコード, 45
 監査ログ全体, 56
標準クリーンアップ, 71

ふ

ファイル vnode トークン, 49, 82
ファイル行の終了記号 \, 65, 67
ファイルのコメント #, 67
ファイルのコメント記号 #, 65, 67
ファイル、ロック, 71, 74
フラグ, 8, 12
 audit_control ファイル行, 12
 audit_user ファイル, 14
 auditconfig コマンドオプション, 40
 概要, 8
 構文, 10, 11
 接頭辞, 10, 12, 36
 定義, 9, 10
 プロセス事前選択マスク, 15
 方針フラグ, 40
 マシン全体, 8, 12
プラス (+) 監査フラグ接頭辞, 10, 12, 36
プロセス監査特性, 16
 監査 ID, 16
 監査セッション ID, 16
 端末 ID, 16
 プロセス事前選択マスク, 15, 24, 25
プロセスグループトークン
 groups トークン, 84, 85
 newgroups トークン, 89
プロセス事前選択マスク
 auditconfig コマンドオプション, 40
 記憶装置容量の縮小コスト, 24, 25
 説明, 15
プロセスの監査特性, 15
フロッピーディスクドライブ
 デバイスクリーンスク립ト, 70

分散システムの auditreduce コマンドの使用, 56

分析, 43, 59
 auditreduce コマンド, 45, 55, 58
 praudit コマンド, 45, 58, 59
 監査機能, 43, 44
 監査レコード形式, 45, 55
 コスト, 23
 ツール, 44, 45

ほ

方針, ix
 auditconfig オプション, 40
 設定, 40
ボンド記号 (#) ファイルのコメント記号, 65, 67

ま

マイナス (-) 監査フラグ接頭辞, 10, 12, 36
マスク、プロセス事前選択
 auditconfig コマンドオプション, 40
 記憶装置容量の縮小コスト, 24, 25
 説明, 15
 マシン全体, 12
マッピング、クラス, 7, 41

み

短い praudit 出力書式, 59
 header トークン, 48

ゆ

ユーザ ID (監査 ID), 6, 16, 44
ユーザが読める監査レコード書式
 監査レコードを変換する, 8, 45
 説明, 45
ユーザが読める書式, ix
ユーザが読める書式に変換, 8, 21
 監査レコードを変換する, 21
ユーザ監査フィールド, 14
ユーザ名
 カーネルイベント, 7
 監査クラス, 9, 10
 監査サーバのマウントポイントのパス名, 32

- 監査フラグ, 9, 10
- ユーザレベルのイベント, 7
- ID
 - auditconfig コマンドオプション, 40
 - 監査, 6, 16
 - 監査セッション, 16
 - 監査セッション, 44
 - 端末, 16
- 監査ファイル
 - 使用, 29
 - 書式, 28
 - タイムスタンプ, 29
 - 閉じられたファイル, 29
 - まだアクティブなファイル, 29
- ユーザレベルイベント
 - 監査レコード, 197
- ユーザレベルのイベント, ix
 - auditconfig コマンドオプション, 39
 - 監査トークン, 46

- 監査レコード, 181
- 説明, 7
- ユーティリティ
 - デバイスの割り当て, 63, 64

ろ

- ログイン/ログアウトメッセージ、1つの
ファイルにコピーする, 57
- ログイン/ログアウトメッセージ、1つの
ファイルに集計される, 57
- ロックファイル
 - 設定, 72
 - 割り当て機構の機能, 72, 74

わ

- 割り当てエラー状態, 64