Sun™ Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング

Release 8.1.0

Sun Microsystems, Inc.
www.sun.com

Part No. 819-2520-10
2005 年 6 月, Revision A

コメントの送付: http://www.sun.com/hwdocs/feedback

原典: *Sun™ Mainframe Transaction Processing Software Troubleshooting and Tuning Guide*
Part No: 817-7437-10
Revision A

目次

図目次

表目次

コード例

はじめに

このマニュアルでは、Sun™ Mainframe Transaction Processing ソフトウェア (Sun MTP) を実行しているときに発生する可能性のある問題について説明し、解決策を示します。また、パフォーマンスを向上させるために、システムを変更する方法についても説明します。

マニュアルの構成

このマニュアルの各章は、発生する可能性のある問題の種類別に構成されています。各章では問題について説明し、問題を解決するための情報を示しています。

第 1 章では、ソフトウェアのアップグレード中に発生する可能性のある問題および初期インストールに伴う問題について説明します。

第 2 章では、Sun MTP の起動を妨げる可能性のある問題について説明します。

第 3 章では、領域とトランザクションの異常終了および回復の方法について説明します。

第 4 章では、クライアント端末に関連する問題について説明します。

第 5 章では、データ破壊の原因と、VSAM ファイルの回復方法について説明します。

第 6 章 では、デバッグツールの使用中に発生する可能性のある問題について説明します。

第 7 章では、非同期で起動されたトランザクションの問題について説明します。

第 8 章では、サン以外のデータベース製品を使用する際に発生する可能性のある問題について説明します。

第 9 章では、標準のバッチ操作に関連する問題について説明します。

第 10 章では、管理サーバーに関連する起動の障害について説明し、デバッグ追跡を有効にする方法についても説明します。

第 11 章では、パフォーマンス上の問題の原因を突き止める方法について説明します。

第 12 章では、Sun MTP 診断ツールについて説明します。

UNIX コマンド

このマニュアルには、システムの停止、システムの起動、およびデバイスの構成などに使用する基本的な UNIX® コマンドと操作手順に関する説明は含まれていない可能性があります。これらについては、以下を参照してください。

- 使用しているシステムに付属のソフトウェアマニュアル
- 下記にある Solaris™ オペレーティングシステムのマニュアル

  `http://docs.sun.com`

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	*machine_name*%
UNIX の Bourne シェルと Korn シェル	$
スーパーユーザー (シェルの種類を問わない)	#

書体と記号について

書体または記号*	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、コード例。	.login ファイルを編集します。 ls -a を実行します。 % You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	% **su** Password:
AaBbCc123	コマンド行の可変部分。実際の名前や値と置き換えてください。	rm *filename* と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 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超える場合に、継続を示します。	% **grep '^#define ** **XV_VERSION_STRING'**
[ ]	省略可能な項目を示します。	unikixmain [-Q]
{ }	垂直線で区切って、代替オプションを示します。	kixfile [-r{Y|N}]
|	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。	EXEC CICS READ DATASET | FILE

* 使用しているブラウザにより、これらの設定と異なって表示される場合があります。

このマニュアルでは、次の書式を使用してコマンドを表記します。

```
$ command required-argument [optional-argument]
```

コマンドが省略可能な引数を持たない場合は、コマンドを入力して Return キーを押します。

関連マニュアル

製品	タイトル	Part No.
Sun Mainframe Transaction Processing ソフトウェア	『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』	819-2514-10
	『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』	819-2515-10
	『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』	819-2516-10
	『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』	819-2517-10
	『Sun Mainframe Transaction Processing ソフトウェア メッセージガイド』	819-2518-10
	『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』	819-2519-10
	『Sun Mainframe Transaction Processing ソフトウェア XA リソースマネージャーの使用』	819-2358-10
	『Sun Mainframe Transaction Processing ソフトウェア ご使用にあたって (Solaris プラットフォーム用)』	819-2521-10
Sun Mainframe Batch Manager ソフトウェア	『Sun Mainframe Batch Manager ソフトウェア 構成ガイド』	819-2505-10
	『Sun Mainframe Batch Manager ソフトウェア インストールガイド』	819-2506-10
	『Sun Mainframe Batch Manager ソフトウェア メッセージガイド』	819-2507-10
	『Sun Mainframe Batch Manager ソフトウェア 移行ガイド』	819-2508-10
	『Sun Mainframe Batch Manager ソフトウェア リファレンスマニュアル』	819-2360-10
	『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』	819-2509-10
	『Sun Mainframe Batch Manager ソフトウェア ご使用にあたって (Solaris プラットフォーム用)』	819-2510-10
Sun Mainframe Administration Tool	『Sun Mainframe Administration Tool ユーザーズガイド』	819-2523-10
高可用性 (HA) エージェント (Sun Cluster 用)	『Sun Mainframe Transaction Processing ソフトウェア 高可用性 (HA) データサービス (Sun Cluster 用)』	819-2522-10
	『Sun Mainframe Batch Manager ソフトウェア 高可用性 (HA) データサービス (Sun Cluster 用)』	819-2511-10
	『Sun Mainframe Security Facility 高可用性 (HA) データサービス (Sun Cluster 用)』	819-2512-10
Sun Mainframe Security Facility	『Sun Mainframe Security Facility 管理者ガイド』	819-2359-10
	『Sun Mainframe Security Facility ご使用にあたって (Solaris プラットフォーム用)』	819-2513-10

製品	タイトル	Part No.
IBM CICS	『CICS アプリケーション・プログラミング・リファレンス』	SC33-1170
	『CICS アプリケーション・プログラミング・ガイド』	SC33-0674
	『CICS Master Index』	SC33-1074
	『CICS Supplied Transactions』	SC33-1686
	『CICS System Programming Reference』	SC33-1689
Server Express	Server Express のマニュアル	*
ACUCOBOL-GT	ACUCOBOL-GT のマニュアル	*
Open PL/I	『Liant Open PL/I User's Guide』	*
	『Liant Open PL/I Language Reference Manual』	*
	『Liant CodeWatch Reference Manual』	*
C	C コンパイラのマニュアル	*
C-ISAM	『C-ISAM Programmer's Manual』	*
	『System Performance Tuning』、Mike Loukides 著、砂原秀樹監訳、株式会社アスキー発行、1995	

* これらのマニュアルは、使用するプラットフォームによって異なります。プラットフォームに該当するマニュアルについては、ご購入先にお問い合わせください。

Sun のマニュアルの注文方法

日本語版を含め、Sun のマニュアルは次のサイトで、表示や印刷、または購入ができます。

http://www.sun.com/documentation

Sun の技術サポート

この製品に関して、このマニュアルでも解決しない技術的な質問がある場合は、次のサイトからお問い合わせください。

http://www.sun.com/service/contacting

コメントをお寄せください

マニュアルの品質改善のため、お客様からのご意見およびご要望をお待ちしております。コメントは下記よりお送りください。

http://www.sun.com/hwdocs/feedback

ご意見をお寄せいただく際には、下記のタイトルと Part No. を記載してください。

『Sun Mainframe Transaction Processing ソフトウェア 障害追跡とチューニング』、Part No. 819-2520-10

第1章

インストール

この章では、Sun MTP をはじめてインストールするとき、またはアップグレードしたり保守用リリースをインストールしたりするときに発生する可能性のある問題について説明します。この章の内容は、次のとおりです。

ライセンスの問題

ライセンスファイルが存在しない

Sun MTP をインストールするには、ソフトウェア使用許可キーを含む有効なライセンスファイルが、`KIXLICDIR` 環境変数に指定されたディレクトリに存在していなければなりません。ライセンスキー自体は、ホスト識別子に関連付けられています。

▼ ライセンスファイルの存在を確認する

1. **領域の設定ファイルに基づいて環境変数を設定します。**
2. **`$KIXLICDIR` ディレクトリに移動します。**
3. **ライセンスファイルが存在していることを確認します。**

   ライセンスファイルは *xxx*`.lic` という名前のファイルです。ファイル名の *xxx* は、ホスト識別子を指します。

4. **ライセンスファイルが存在しない場合、次のいずれかの手順を実行します。**
   - 『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』の指示に従って、`$KIXLICDIR` ディレクトリにライセンスファイルを作成します。
   - ライセンスファイルを `$KIXLICDIR` 以外のディレクトリにインストールしていないかどうかを確認し、インストールしていた場合は、`$KIXLICDIR` をその新しいディレクトリに再設定して、領域の起動を試みます。

ライセンスファイルエラーメッセージが表示される

ソフトウェア使用許可キーが失効していたり、ファイルに正しく入力されなかった場合、Sun MTP は起動時にエラーメッセージを表示します。ご購入先に連絡して新しいソフトウェア使用許可キーを入手するか、失効している場合は現在のソフトウェア使用許可キーの期間を延長する必要があります。問題を連絡する際には、必ずシステムのホスト識別子と現在のソフトウェア使用許可キーをご用意ください。

新しい使用許可キーでこの問題が発生した場合は、ライセンスファイルに情報を誤って入力した可能性があります。大文字と小文字で正しい文字数分入力していることを確認します。

初期インストール

Sun MTP をはじめてインストールするときは、『Sun Mainframe Transaction Processing ソフトウェア インストールガイド』の手順に従ってください。エラーが表示された場合は、その状態と正確なエラーメッセージをご購入先までご連絡ください。

ディスク容量の不足

Sun MTP のインストールには十分なディスク容量が必要です。さらに、インストール中に作成される `unikixtran` や `unikixvsam` などの Sun MTP オブジェクトを構築するためのディスク容量を必要とします。これらのオブジェクトの構築で問題が発生した場合は、`kixinstall` 構築プロセス中に `/tmp/kixinstall.log.`*pid* ファイル内に生成されるディスク領域エラーメッセージを探します。*pid* は、`kixinstall` プロセスのプロセス識別子 (ID) です。

ディスク容量の要件については、リリースノートを参照してください。

kixinstall のエラーメッセージ

kixinstall ユーティリティーを使用して Sun MTP 実行可能ファイルを構築する途中でエラーメッセージが表示される場合は、次の点を調べてください。

- /tmp/kixinstall.log.*pid* には、unikixtran のエラーをはじめとする構築エラーが記録されています。*pid* は、kixinstall プロセスのプロセス ID です。kixinstall プログラムの終了時には、このファイル名の入ったメッセージが表示されます。さらに支援が必要な場合は、kixinstall.log.*pid* ファイルをご購入先に電子メールで送信してください。
- Server Express を使用している場合は、$COBDIR がインストールディレクトリを指し、$COBDIR/bin が COBOL 実行可能ファイルのディレクトリを指すように設定し、これらのディレクトリが $PATH 内にあることを確認してください。
- ACUCOBOL-GT® を使用している場合は、$ACUCOBOL がインストールディレクトリを指し、$ACUCOBOL/bin が COBOL 実行可能ファイルのディレクトリを指すように設定し、これらのディレクトリが $PATH 内にあることを確認してください。
- 共有ライブラリ環境変数 (Solaris では LD_LIBRARY_PATH、AIX では $LIBPATH) を正しいライブラリディレクトリに設定していることを確認します。次に例を示します。

  Server Express 環境では、次のディレクトリも含める必要があります。
  $UNIKIX/lib:$COBDIR/lib

  ACUCOBOL-GT 環境では、次のディレクトリも含める必要があります。
  $UNIKIX/lib:$ACUCOBOL/lib

- Oracle® などサン以外のデータベースを使用している場合は、適切な変数を設定していることを確認します。必要な変数のリストについては、$UNIKIX/examples/*databasename*/*language* ディレクトリにある設定ファイルを参照してください。『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』も参照してください。

保守インストール

Sun MTP の保守用リリースをインストールする場合は、先にすべての領域を停止してから、初期インストールと同じ手順を行います。

Sun MTP の現在のリリースの上に、保守用リリースをインストールしないでください。その代わりに、たとえば `/mtp/`*nnnnnn-nn*`/MTP8.1.0p1` のように、リリース番号をディレクトリ名の一部に使用して、ほかのディレクトリにインストールします。ここで *nnnnnn-nn* は、パッチ番号です。通常、保守用リリースにはパッチノートに記録されている変更だけが含まれ、2、3 のモジュールだけに影響を与えます。しかし、提供されるのは変更されたモジュールだけではなく、パッケージ全体です。上記の手順を行うことにより、インストールするモジュールの選択に関する混乱が最小限になり、保守用リリースが問題を起こした場合でも元のリリースに戻しやすくなります。パッチのリリースをインストールする手順については、パッチの Readme ファイルを参照してください。

第2章

起動

この章では、Sun MTP の起動を妨げる可能性のある問題と解決策について説明します。この章では、次のトピックについて説明します。

Sun MTP を起動できない

Sun MTP を起動しようとすると、エラーメッセージ「`shmat errno=22`」が表示され、領域が起動しません。

共有メモリー要件の計算については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。`unikixmain` (`kixstart`) のオプションについては、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

メモリー不足

システムの実メモリーが十分であるにもかかわらず、メモリー不足のために領域が起動しません。メモリー構成のガイドラインについては、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

カーネルパラメータの誤り

システムが起動しない場合、カーネルパラメータが誤って設定されている場合があります。オペレーティングシステムのパラメータについては、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

構成を変更すると回復が失敗する

トランザクション処理プログラムの合計数が、たとえば 10 個から 8 個など少ない値に変更されたあと、すでに存在しないトランザクション処理プログラムに対する未確定の処理単位 (UOW) で領域が失敗する場合、領域は回復処理を完了できず、領域の起動は強制的に中止され、次のエラーメッセージが表示されます。

KIX4801E XA: Transaction processor configuration has changed. Region cannot be started.

KKIX4806E XA: Configured transaction processors is *%d*. Recovery requires *%d* as a minimum.

トランザクション処理プログラムの数を再構成して、回復処理を完了できるようにする必要があります。構成テーブル `vct.tbl` の以前のバージョンを使用できる場合は、それを使って構成をリセットすると、領域を再起動できます。以前の構成を使用できない場合は、領域の構成を変更して、回復処理を完了できるようにする必要があります。

▼ トランザクション処理プログラムの数をリセットする

1. `$KIXSYS` ディレクトリに移動します。
2. `kixexptbl` ユーティリティーを実行して、既存の領域構成ファイル (システムテーブル) を、それぞれが関連付けられている `.lst` ファイルにエクスポートします。

   `vct.lst` ファイルには VCT 構成が入っていて、その中にトランザクション処理プログラムの合計数も含まれています。

3. **テキストエディタで、`vct.lst` ファイルを開きます。**

   トランザクション処理プログラムの合計数が、コンマ区切りのリストのフィールド 8 に定義されています。たとえば、次の `vct.lst` ファイルには、トランザクション処理プログラムの合計数として 8 という値があります。

```
SYSA_RECOVERY ,KIXDATA        ,,200,0,128,32,8,32,1,1,10,2,N,
```

4. **フィールド 8 の値を、エラーメッセージ KIX4806E に指定された最小値以上の値に変更します。**

5. **`kiximptbl` ユーティリティーを実行して、`vct.lst` ファイルとその新しい構成をインポートします。**

6. **領域を起動します。**

   これで回復処理を完了できます。

第3章

領域とトランザクションの異常終了

この章では、領域とトランザクションの異常終了の診断および処理に関する情報を提供します。次のトピックが含まれます。

異常終了

この節では異常終了の原因について説明します。詳細は、12 ページの「領域の異常終了後の再起動」も参照してください。

Sun MTP のクラッシュ

Sun MTP の異常終了は、正しい手順の中で発生する場合と、「不正終了 (abend)」とも呼ばれる突発的な中止によって発生する場合があります。正常に終了した場合は、次のイベントが発生します。

- すべてのプロセッサが終了する
- `kixsnap` ユーティリティーが自動的に実行される
- プロセッサ間の通信に使用されるすべての共有メモリー、セマフォー、およびキューが解放される

終了が正常に行われなかった場合は、`kixsnap` ユーティリティーを手動で実行してください。

注 – 強制的な中止処理によって端末がロックされたために、新しい端末を使用してログオンする場合、kixsnap を実行する前にすべての Sun MTP 環境変数 ($KIXSYS を含む) をリセットし、$UNIKIX/bin がパスに存在することを確認する必要があります。

kixsnap ユーティリティーは、$KIXSYS/debugkix ディレクトリまたは KIXSNAPDIR 環境変数で定義したディレクトリに圧縮ファイルを作成します。snapshot.*mmdd_hhmmss* ファイルには、ユーザーとプロセス、異常終了の時刻などシステム全体に関する情報が収められます。

注 – 領域がバッチノードに関連付けられている場合は、kixsnap ユーティリティーによって Sun MBM の ebmsnap ユーティリティーも実行されます。ebmsnap ユーティリティーについては、『Sun Mainframe Batch Manager ソフトウェア ユーザーズガイド』を参照してください。

kixsnap の出力は解釈が困難な場合があるので、ファイルを保存してご購入先に連絡してください。サポート担当の技術者が、スナップショット情報を電子メール、郵便、または ftp のいずれかで送付していただくようお願いします。

kixsnap の詳細は、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

アプリケーショントランザクションの強制的な中止

アプリケーショントランザクションの強制的な中止の原因となるエラーには、次の 2 種類があります。

- エラーがそれまでに HANDLE コマンドまたは IGNORE コマンドで処理されていないために、EXEC CICS コマンドによって返される例外条件
- アプリケーションの実行時コードによって検出されるエラー

処理されていない EXEC CICS コマンドの例外条件によってトランザクションが強制的に中止し、デバッグモードが動作していない場合、画面の 23 行目にメッセージが表示されます。このメッセージには、トランザクション ID、EXEC CICS コマンド名、および例外条件の名前が含まれます。次に例を示します。

```
Tran KTBM has abended:Function WRITEQ TD Condition QIDERR
```

たとえば、プログラム管理テーブル (PCT) で指定されているが、ディスクで見つからなかったプログラムまたはマップセットのように、対応するリターンコードのない条件が強制的な中止を発生させることもあります。この場合、画面の 23 行目にテキストメッセージが表示されます。`$KIXSYS` ディレクトリの `unikixmain.err` ファイルに、強制的な中止の相対アドレスなど、追加情報が見つかることがあります。強制的に中止されたプログラムのリストファイルで、アドレスが見つかる場合もあります。Sun MTP は、新しいトランザクション処理プログラムを起動して、強制的に中止されたプログラムと置き換えます。

さらに、書式付きアプリケーションダンプ機能が有効な (デフォルトの条件) 場合、Sun MTP は、このトランザクションの名前で書式付きダンプファイルを生成します。70 ページの「ダンプ機能の有効化と無効化」と 77 ページの「トランザクションが不正終了した位置の特定」も参照してください。

システムがデバッグモードで動作し、トランザクションの処理中に強制的な中止が発生した場合、エラーメッセージがデバッグ機能画面に表示されます。図 3-1 に強制的な中止の例を示します。画面中央の情報は、エラー状態とエラーが発生したプログラムを示します。

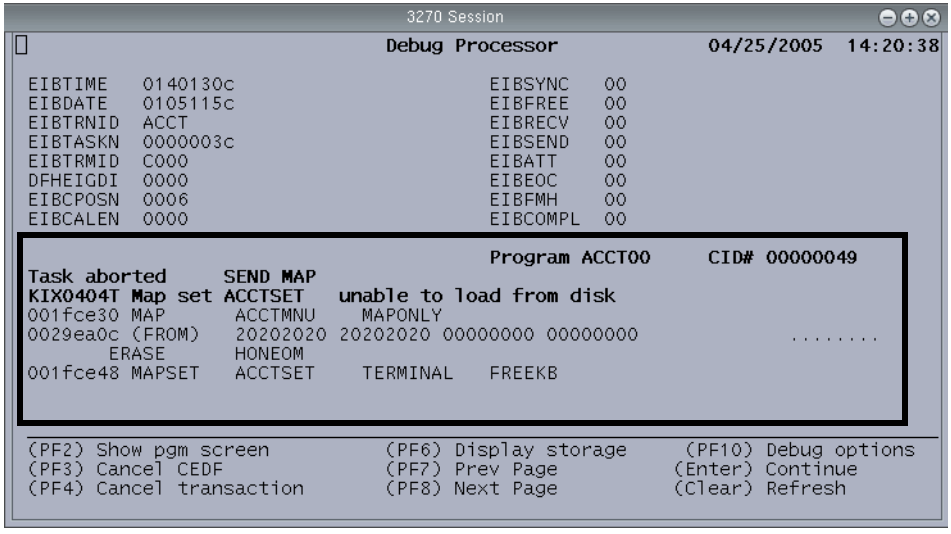

図 3-1 デバッグ機能画面 - 強制的な中止エラー

▼ デバッグ機能で強制的な中止から回復する

1. **PF4 キーを押します。**

   取り消しの確認を求める画面が表示されます。

2. **Enter キーを押してトランザクションを取り消すか、PF3 キーを押してデバッグ機能に戻ります。**

   - トランザクションを取り消すと、画面は強制的な中止時の状態に戻ります。
   - その画面がトランザクションを入力できる画面であれば、トランザクションを入力できます。それ以外の場合は、Clear キーを押して、画面をクリアします。

領域の異常終了後の再起動

領域が異常終了した場合は、クリーンアップ操作を行い、この領域の VSAM ファイルの構造を確認します。これを実行しないと、新しい Sun MTP サーバーを起動できない場合があります。

注意 – 同じユーザー ID を使用して起動した複数の領域を実行中の場合は、ほかの領域を停止してから次の手順を行ってください。そうしなければ、適切に動作している領域に関連付けられた IPC リソースを削除してしまうおそれがあります。

▼ 異常終了後にクリーンアップする

1. **シェルウィンドウを開いて、終了した領域の設定ファイルを実行します。**
2. **プロンプトで、`kixclean -a` コマンドを実行して、終了プロセスを完了します。**

3. **ipcs コマンドを実行して、すべてのプロセス間通信 (IPC) リソースが解放されていることを確認します。**

   このコマンドは、ホストで使用中のすべてのセマフォー、キュー、および共有メモリーを表示します。IPC リソースを使用するほかのアプリケーションが動作している場合、ipcs からの出力の読み込みが困難になることがあります。次のコマンドを使用して、領域を起動したユーザーのユーザー ID に関連付けられた IPC リソースだけを表示します。

```
$ ipcs | grep userid
```

   IPC リソースがまだ存在する場合は、ipcrm コマンドを使用して削除します。同一のユーザー ID が複数の領域を持っている場合、複数のエントリが存在する可能性があります。この場合、タイプ D (defunct/dead) のエントリを削除します。ipcrm(1) コマンドのマニュアルページを参照してください。

4. **kixverify -r ALL を実行して、VSAM クラスタのアクティビティーカウントをリセットします。**

   kixverify を実行しない場合、Sun MTP 領域は次のようなメッセージを表示してから、起動中に強制的に中止します。

```
Error accessing Catalog, file is open to another Sun MTP system
currently or it was not closed properly
Activity count error on cluster %s, override status to deferred
open
```

5. **kixstart を実行して領域を再起動し、ロールバックを実行可能にします。**

▼ ファイルの完全性を確認する、または VSAM ファイル構造の妥当性を検査する

1. **領域を再起動し、それによってロールバックの実行が可能になってから、kixstop コマンドを使用して領域を停止します。**
2. **kixvalfle を実行して、クラスタの VSAM ファイルの構造を検査します。**
3. **kixvalfle が問題を報告する場合は、kixsalvage を実行します。これによって、すべての修復可能なデータはスキャンされ、新しいファイルにダンプされます。**

   コマンドとすべてのオプションの詳細は、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。また、データの破壊と回復については、第 5 章も参照してください。問題が継続する場合は、ご購入先に連絡してください。

第4章

クライアント端末

この章では、クライアント端末のもっとも一般的な問題について説明します。次のトピックが含まれます。

領域が起動しても通信サーバーが起動しない

起動中に `Writing snapshot` メッセージが表示され、`unikixmain.err` ファイルに次のメッセージが含まれる場合

```
ld.so.1:unikixCommMgr:fatal:libACE.so:open failed:No such file or directory
```

`LD_LIBRARY_PATH` 環境変数には、`$UNIKIX/lib` ディレクトリのエントリが含まれていません。この場合、領域は起動しますが、通信マネージャープロセス (`unikixCommMgr`) は起動しません。したがって、通常は起動する通信サーバーが起動しません。TN3270 エミュレータ、WebSphere MQ、または SNA システム間通信を通じて領域に接続できません。

問題を修正するには、領域の設定ファイルに $UNIKIX/lib が含まれるように $LD_LIBRARY_PATH 環境変数を変更し、この設定ファイルを指定して領域を再起動します。

TN3270 セッションが接続しない

PC ベースの TN3270 クライアントに関連する TN3270 のもっとも一般的な問題は、領域に接続できないことです。

この問題が発生した場合は、次の作業を実行してください。

- TN3270 サーバー製品の使用許諾を受けていることを確認します。
- クライアントが適切なホストとポート番号に対して構成されていることを確認します。
- unikixtnemux プロセスによってポートが正常に開いていることを確認します。ポートが開かない場合は、そのポートがほかのプロセスに占有されていることを示す次のようなエラーが unikixmain.log ファイルと unikixmain.err ファイルに記録されています。

```
02/21/2005 14:00:17 TNmux.8299 :KIX2530E Unable to acquire port 7021, MTP must
be recycled after the conflict is resolved.
```

ポートの重複を解決します。別のポート番号を $KIXSYS/unikixrc.cfg ファイルに指定する必要がある場合もあります。重複を解決してから、領域を再起動します。

端末またはキーボードがローカルの Sun MTP クライアントで動作しない

この節では、terminfo データベースを使用して端末の問題を解決する方法について説明します。

terminfo 定義の作成とインストール

tic コマンドは、Sun MTP 領域で使用する、カスタマイズされた端末の terminfo 定義を作成します。領域で使用する、機能拡張された terminfo 定義が提供されている場合、terminfo ファイルの再定義は、$UNIKIX/etc ディレクトリにあります。再定義されたものは、ファイル名の接尾辞「.tic」で識別できます。これらのファイルによって、Sun MTP の画面では、最上部ではなく、最下部に状態行が表示されます。

▼ tic コマンドを使用して標準的な terminfo ディレクトリ構造に端末定義を追加する

1. **/usr/lib/terminfo にある、システムの terminfo データベースを変更するには、root アクセス権が必要です。**

   たとえば、サイトで X サーバーの端末が使用されている場合、root つまりスーパーユーザーとしてログインしてから、次のように入力します。

```
$ cd $UNIKIX/etc
$ tic xterms.tic
```

   root としてログインできない場合は、UNIX システム管理者に連絡してください。

2. **端末タイプの tic ファイルが存在しない場合は、infocmp コマンドまたは untic コマンドを使用して作成できます。次に例を示します。**

```
$ infocmp /usr/lib/terminfo/x/xterms > xterms.tic
```

3. **作成した tic ファイルを編集して、tic コマンドでコンパイルします。**

```
$ tic xterms.tic
$ exit
```

4. **root アクセス権を必要としない独自の** terminfo **データベースを作成するには、** TERMINFO **環境変数がそのディレクトリを指すように設定します。次に例を示します。**

```
$ TERMINFO=$UNIKIX/terminfo; export TERMINFO
$ tic custom-termname.tic
```

tic コマンドは、terminfo ファイルをコンパイルして $UNIKIX/terminfo/c/*custom-termname* にデータベースを作ります。terminfo ディレクトリ階層に作成されるファイルの名前は、tic ファイルのコメントでない最初の行の最初のフィールドから取得されます。

独自の定義を使用するには、領域を起動する前に TERMINFO 環境変数を設定するか、新しい端末の名前を設定する unikix -k オプションを使用して領域を起動します。

変更した terminfo の記述は、$UNIKIX/local/lib に格納しておくと、上書きされることなく、オペレーティングシステムの今後のリリースに適用できます。

システムが 1 つの領域だけで動作する場合、供給されたファイルを編集して、構成を簡略化できます。ただし、オペレーティングシステムソフトウェアの次のバージョンをインストールすると、これらの変更内容が消失することがあります。したがって、標準ファイルを変更した場合は、$UNIKIX/local ディレクトリ構造にコピーを保存します。また、編集エラーから回復できるように、編集する前に元のファイルをコピーする必要があります。

terminfo によるアプリケーションキーパッドモードの変更

端末定義に curses や terminfo を使用するアプリケーションでは、終了前に端末リセットまたは初期設定が行われる場合があります。この結果、アプリケーションの実行前は有効になっていた端末の設定が、アプリケーションの実行前と異なる不定の条件に設定されることがあります。残念ながら、端末の以前の設定を記録し、復元する機構はありません。

たとえば、領域から vi を呼び出す前にアプリケーションキーパッドが有効になっていた場合は、vi の終了時にリセットされます。ユーザーには、キーパッドのアテンション ID (AID) キーが機能しなくなったか、キーによって不正なシーケンスが生成されているように見えます。

最良の方法は、この TERM の値の terminfo 定義を操作時のアプリケーションキーパッドモードにリセットしないことです。infocmp コマンドを使用して、terminfo 定義の現在の内容を取り込んで、調査することができます。アプリケーションキーパッドモードのリセットを含むエントリを変更し、tic コマンドを使用して、変更内容を terminfo カタログに適用します。infocmp および tic の使い方については、17 ページの「terminfo 定義の作成とインストール」を参照してください。

端末がアプリケーションキーパッドモードを開始および終了するシーケンスは、端末によって異なります。VT220 端末では、\E> でこのモードを解除し、\E= で設定します。このシーケンスは、-is2、-rmkx など、複数の terminfo エントリで発生します。さらに、terminfo ファイルは、端末タイプが同じでもベンダーが異なる場合や、ベンダーが同一でも製品ラインが異なる場合、terminfo 定義が異なることがあります。特定の端末のシーケンスについては、ご購入先の技術資料を参照してください。

terminfo の値の検証

terminfo の値が COBOL のソースデバッガに適合していることを確認するには、デバッガを使用しているシステムを見つけるか、デバッガとともに機能する terminfo エントリの tic ファイルのコピーを取得して、使用している tic ファイルの値と比較します。詳細は、17 ページの「terminfo 定義の作成とインストール」を参照してください。

tic ファイルでは、端末名が、コメントを除く最初の行の最初のフィールドとして含まれ、そのあとにさまざまなキーの定義が続きます。たとえば、作業ファイルで上向きの矢印キーの値が \EOD (キーワード kcuu1=\EOD) に等しい場合、使用している tic ファイルでも同じである必要があります。

表 4-1 に、tic ファイルに含まれている、アプリケーションモードとカーソルモードでの矢印キー (デバッガの正常な操作に必要) のキーワードを示します。デバッガではキーがカーソルモードで機能することを想定しています。マニュアルページで terminfo を表示すると、すべてのキーワードが一覧されます。

表 4-1　キーボード定義のキーワード

定義	キーワード (アプリケーションモード)	キーワード (カーソルモード)
上向きの矢印キー	kcuu1	cuu1
左の矢印キー	kcub1	cub1
右の矢印キー	kcuf1	cuf1
下向きの矢印キー	kcud1	cud1
ファンクションキー F1 ～ F12	kf1-kf12	

▼ 特定の terminfo 定義を使用する

- **curses を使用するクライアントを起動する前に、terminfo の名前を TERM 環境変数に設定します。**

  これは、ユーザーの .profile ファイルまたは .login ファイル (TERM=*term* または setenv TERM *term*) の中で行うか、xterm に -tn オプションを付けて指定します。シェルコマンドを実行して、TERM 環境変数を設定することもできます。

数字入力キーがアプリケーションモードにならない

Sun MTP では、キーボードの数字入力キーがアプリケーションモードである必要があります。このモードでは、マイナスキーがキーボードリセットを実行し、プラスキーが画面クリアを送信します。ただし、強制的な中止後、端末が数値モードのままになっていることがあります。この状況からの回復を容易にするには、キーボードを 1 つのモードからほかのモードにリセットするスクリプトファイルを作成します。

以下の 2 つの手順は、次の機能を持つスクリプトファイルの作成方法を示します。

- キーパッドをアプリケーションモードにリセットする
- キーパッドを数値モードにリセットする

これらの手順では、VT100、VT220、または VT320 モードの端末のためのキーシーケンスが示されます。異なるタイプの端末を使用している場合は、端末のマニュアルでリセットシーケンスを参照してください。

▼ キーパッドをアプリケーションモードにリセットするスクリプトを作成する

1. **vi または任意のテキストエディタで、appl という名前 (または任意の名前) のファイルを作成します。**
2. **スクリプトファイルのテキストを入力します。**

   a. **echo という単語をタイプし、次にスペースを加えます。**

   b. **Shift キーと数字の 6 を押して、ファイルにキャレット文字 (^) を挿入します。**

   c. **[文字をタイプします。**

   d. **等号記号 (=) をタイプします。**

e. Escape キーを押して、挿入モードを終了します。

ファイルの内容は、次のようになります。

```
echo ^[=
```

3. ファイルを保存し、エディタを終了します。
4. ファイルを実行可能にしてから、ファイルを実行し、キーパッドをアプリケーションモードにリセットします。

```
$ chmod 777 appl
$ appl
```

▼ キーパッドを数値モードにリセットするスクリプトを作成する

注 – 数値モードではマイナスキーはマイナス記号を、プラスキーはプラス記号を生成します。

1. vi または任意のテキストエディタで、num という名前 (または任意の名前) のファイルを作成します。
2. スクリプトファイルのテキストを入力します。

   a. echo という単語をタイプします。

   b. Shift キーと数字の 6 を押して、ファイルにキャレット文字 (^) を挿入します。

   c. [文字をタイプします。

   d. 大なり記号 (>) をタイプします。

   e. Escape キーを押して、挿入モードを終了します。

   ファイルの内容は、次のようになります。

```
echo ^[>
```

3. ファイルを保存してエディタを終了します。

4. ファイルを実行可能にしてから、ファイルを実行し、キーパッドを数値モードにリセットします。

```
$ chmod 777 num
$ num
```

COBOL デバッガで矢印キーがカーソルモードにならない

COBOL デバッガを正しく操作するには、矢印キーがカーソルモードであることが必要です。これは通常操作の場合です。ただし、Sun MTP またはデバッガが異常終了すると、カーソルキーがアプリケーションモードのままになっている場合があります。

矢印キーをカーソルモードにリセットするスクリプトを作成できます。この処理の手順では、VT100、VT220、または VT320 モードの端末のためのキーシーケンスを示します。異なるタイプの端末を使用している場合は、端末のマニュアルでリセットシーケンスを参照してください。

▼ 矢印キーをカーソルモードにリセットするスクリプトを作成する

1. `vi` または任意のテキストエディタで、`curs` という名前 (または任意の名前) のファイルを作成します。
2. スクリプトファイルのテキストを入力します。
   a. `echo` という単語をタイプします。
   b. Shift キーと数字の `6` を押して、ファイルにキャレット文字 (^) を挿入します。
   c. ファイルに `[` 文字をタイプします。
   d. 次の文字をタイプします。`[?1l`
   e. Escape キーを押して、挿入モードを終了します。
   ファイルの内容は、次のようになります。

```
echo ^[[?1l
```

3. ファイルを保存してエディタを終了します。

4. ファイルを実行可能にしてから、ファイルを実行して、矢印キーをカーソルモードにリセットします。

```
$ chmod 777 curs
$ curs
```

矢印キーをアプリケーションモードにリセットするスクリプトを作成するには、上記の手順に従いますが、ファイルの内容を次のように変更します。

```
echo ^[[?1h
```

起動したユーザー ID 以外から Sun MTP に接続できない

Sun MTP が特定のユーザー ID から起動され、ユーザー ID の異なるほかのユーザーが領域に接続できません。

▼ この問題を解決する

1. `KIXSYS` 環境変数が正しく設定されていることを確認します。

2. 少なくとも `$KIXSYS` ディレクトリの読み取り許可を持っていることを確認します。

3. 起動時に `unikixmain` に `-Q` オプションを補う必要があるかどうかを判定します。

   `unikixmain` の起動オプションについては、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

4. `unikixl` モジュールと `unikixstrt` モジュールが root に所有されるよう設定され、setuid ビットがセットされていることを確認します。

   通常、この 2 つのモジュールは `$UNIKIX/bin` ディレクトリにあります。

たとえば、unikixl および unikixstrt モジュールの所有者とアクセス権を変更するには、次のコマンドを入力します。

```
# chown root unikixl
# chmod 4755 unikixl
# chown root unikixstrt
# chmod 4755 unikixstrt
```

Sun MTP クライアント端末が異常終了する

次の場合、ユーザーの Sun MTP クライアントが異常終了することがあります。

- ユーザーがプロセスを異常終了させる
- クライアントプロセスが強制的に中止される
- オペレーティングシステムに問題がある

UNIX クライアントは curses を使用するため、クライアントが異常終了すると、端末の記述が通常と異なる状態になっていることがあります。 stty コマンドを使用して、これを修正できます。このプロセスを自動化するには、次の文を各ユーザーの .profile に追加します。

```
stty -g > $HOME/.stty
sane(){
       stty `cat HOME/.stty`
}
```

最初の文によって、ログイン時の端末の状態が保存されます。2 番目の文は、必要に応じてユーザーが使用できるコマンド sane を設定します。このコマンドは、端末をログインの状態にリセットします。異常終了後、ユーザーは sane を入力し、次に Ctrl-J を押して、正常な状態に戻ることができます。Return キーは動作しないことがあるので、Ctrl-J を使用する必要があります。

端末が正常な状態に戻ったあと、ユーザーは再び領域にログインできます。

第5章

VSAM ファイルの解析

VSAM ファイルの内容が妥当かどうかを確定するために、解析が必要になることがあります。解析の支援には Sun MTP の `kixvalfle` ユーティリティーを使用できます。`kixvalfle` は、物理ブロックの関係を解析し、起こりうる問題点を報告します。

この章では、次のトピックについて説明します。

また、VSAM データセットでのレコードの位置の特定方法についても説明します。ファイルの完全性については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

データの破壊

この節では、データセットが破壊されているかどうかを確かめる方法、破壊されたデータセットを回収する方法、および回収したデータセットをロードする方法について説明します。

▼ 破壊されたデータセットを特定する

1. **領域を停止するか、疑わしいデータセットを閉じて、アクセスできないようにします。**
2. **疑わしいデータセットについて `kixvalfle -i` コマンドを実行します。**

   出力ファイルをファイルにリダイレクトします。そのファイルをご購入先に送ってさらに詳しく解析できます。次に例を示します。

```
$ kixvalfle -i dataset > techsupportfile
```

3. **ブロックの欠落がレポートされた場合、または「no errors」が示されない場合は、データセットを回収する必要があります。**

▼ 破壊されたデータセットを回収する

1. **領域を停止するか、疑わしいデータセットを閉じて、アクセスできないようにします。**
2. **破壊されたデータセットについて `kixsalvage` コマンドを実行します。**

   `-c` オプションを使用して Micro Focus ソートユーティリティーでソートできる `mfrcdv` 形式のファイルを作成するか、`-2` オプションを使用してほかのソートユーティリティーでソートできる `recordv` 形式のファイルを作成できます。次に例を示します。

```
$ kixsalvage -c dataset
```

3. **ソートユーティリティーを使用して、回収したデータをソートします。**

   データセットの定義、レコード長、およびキーの属性が必要です。

▼ 回収されたデータセットをロードする

1. **領域を起動します。**

   データセットに対する読み取り許可と書き込み許可を持っていることを確認します。

2. **次のタスクのいずれかを実行します。**

   - Sun MTP 標準バッチを使用する場合

     a. **テキストエディタを使用して、次のコマンドを含むバッチシェルスクリプトを作成します。**

```
kixfile -lY dataset
unikixbld -i -d dataset
unikixbld -tv -d dataset -s salvagedfile -r recordformat
kixfile -lN dataset
```

     回収されたファイルの名前とレコードの形式は、`kixsalvage` コマンドの実行時に指定した値と一致する必要があります。

     b. **スクリプトを保存します。**

   - Sun MBM を使用する場合、JCL スクリプトからファイルの `delete define` を実行するか、`unikixbld` ユーティリティーを使用して、ファイルを初期化します。

3. **`unikixbld` または JCL から `IDCAMS REPRO` プロシージャーを使用して、回収されてソート済みのファイルをデータセットにロードします。**

4. **ジョブスクリプトをサブミットします。**

VSAM のブロック構造

Sun MTP の各 VSAM ファイルは、表 5-1 に示すブロック構造を使用します。

表 5-1　ファイルブロックのレイアウト

オフセット	長さ (バイト)	フィールド名	フィールドの説明
0	4	Activity count	ファイルの検証に使用されます。データファイルのアクティビティーカウントが、索引ファイルのアクティビティーカウントに等しくない場合、ファイルは開きません。
4	4	Next block number	ファイル内でキーシーケンスの次にあるブロックを指します。ファイル内でキーシーケンスの最後にあるブロックの場合、このフィールドはゼロになります。このフィールドは、ESDS (入力順) および RRDS (相対レコード) データセットには使用されません。
8	2	Data length	ブロックで使用されるバイト数。20 バイトのヘッダーは含まれません。4K バイトのブロックサイズを使用する VSAM ファイルの場合、このフィールドは 0 ～ 4076 の値を取ります。このフィールドは、RRDS ファイルには使用されません。
10	1	Block type Block size	上位の桁 (太字) は次の値のいずれかになります。 0x**00**　データブロックを指します。索引ファイルでのみ使用されます。 0x**20**　索引ブロックを指します。索引ファイルでのみ使用されます。 0x**40**　データブロック。データファイルでのみ使用されます。 0x**60**　空きブロック。索引ファイルとデータファイルの両方で使用されます。 下位の桁 (イタリック体) はブロックサイズを示します。 0x0*0*。数値の意味は次のとおりです。 0 = 4K バイト 1 = 8K バイト 2 = 16K バイト 3 = 32K バイト
11	1	Segment indicator	物理ファイルのセグメントの合計とセグメント番号。スパン可能な唯一のファイルである KSDS (キーシーケンス) データファイルにだけ設定できます。 セグメントの合計は上位の桁、セグメント番号は下位の桁です。これらは、0 ～ 7 の値を取り、1 ～ 8 個のファイルを表します。たとえば、「0x73」は、このファイルが 8 ファイルのスパンファイルセットのうちの 4 番目のファイルであることを意味します。

表 5-1　ファイルブロックのレイアウト (続き)

オフセット	長さ (バイト)	フィールド名	フィールドの説明
12	4	Last block in the file	ファイル内の最後の物理ブロックを指します。ファイルの最初のブロック、またはスパンファイルの最初のセグメントの最初のブロックでのみ有効です。
16	4	First unused block	最初の未使用ブロックのブロック番号。個々の未使用ブロックは、「Next Block Number」フィールドでチェーン状にリンクされます。
20	4076, 8172, 16364 または 32748	Record data	1 つまたは複数のデータレコードがこの領域に入ります。長さは、VSAM ブロックサイズ設定 (`unikixmain -b` オプション) によって決定されます。

ESDS ファイルでのレコードの位置の特定

ESDS ファイルは、レコードを順番に格納します。この種類のファイルでは、レコード ID (RIDFLD) フィールドから、レコードの物理的な位置を算出できます。

▼ レコード長が可変の ESDS ファイルでレコードの位置を算出する

1. VSAM ブロックサイズから 20 バイトを引いた値で RIDFLD を割って、1 を加えることによって、ブロック番号を求めます。
2. 手順 1 の結果にブロックヘッダーの長さ (20) を加えて、レコードヘッダーの位置を特定します。
3. レコードヘッダーの長さ (4) を加えて、レコードデータの位置を特定します。

▼ レコード長が固定の ESDS ファイルでレコードの位置を算出する

1. RIDFLD をレコード長で割り、ゼロを基準とするレコード番号を算出します。
2. レコードサイズにレコードヘッダーサイズを加えた値で、各ブロックで使用可能なスペース (4076、8172、16364、または 32748) を割ることによって、ブロックに入るレコードの数を算出します。
3. レコード番号をブロック内のレコードの数で割り、1 を加えて、ブロック番号を算出します。

   この結果、ブロックのレコード番号が得られます。
4. ブロックのレコード番号に、レコードサイズとレコードヘッダーのサイズ (4) を掛けて、データ領域のレコードヘッダーの位置を特定します。
5. ブロックヘッダーの長さ (20) を加算して、ブロック内のレコードヘッダーの位置を特定します。

   ESDS ファイルは、レコードエディタを使用して読むことができます。あるいは `kixview` を使用して表示できます。レコードエディタについては『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を、`kixview` については『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

RRDS ファイルでのレコードの位置の特定

RRDS ファイルも、レコードを順番に格納します。この種類のファイルでは、レコードの物理的な位置を RIDFLD から算出できます。ブロックに入るレコードの数を計算し、レコードを含むブロックを特定してから、そのブロック内部のレコードの位置を算出することによって、レコード番号をバイト位置に変換できます。

VSAM ブロックサイズは、4K、8K、16K、または 32K バイトです。次の手順では、ブロックサイズはデフォルトの 4K バイト、レコードサイズは 80、RIDFLD は 410 を使用します。

▼ レコードが始まるファイルのオフセットを算出する

1. **ブロックごとのレコード数は、VSAM ブロックサイズから 20 バイトを引いた数 (4076) を、レコードあたりのバイト数 80 とレコードヘッダーの 4 バイトを加えた数で割ったものと等しくなります。**

   つまり、各ブロックに 48 レコードが含まれ、各ブロックの終わりに未使用の 44 バイトが残ります。

2. **ブロック番号は、RIDFLD の 410 から 1 を引いた数を (RRDS RIDFLD は 1 を基準とするため)、ブロックあたりのレコード数 48 で割って 1 を加えた数と同じです (ブロック番号も 1 を基準にする)。**

   これにより、ブロック番号は 9 と算出されます。

3. **各ブロックは 4096 バイトです。**

   したがって、9 番目のブロックは、ファイルの 32768 バイト目から始まります (0 が基準)。

4. **ブロック内のオフセットを算出するには、ブロック内のレコード番号を求めます。**

   これは、RIDFLD から 1 を引き (1 を基準にするため)、この数字からブロックあたりのレコード総数とブロック番号の積を引いた数字と等しくなり (410 - 1 - (48 * (9-1)))、レコード番号 25 が算出されます。ブロックのオフセットは、ブロックヘッダーサイズ、合計レコードサイズとブロックのレコード番号の積、ブロックヘッダーサイズの合計 (20 + (25*84) + 4) に等しくなり、ブロックのオフセット 2124 が得られます。

5. **ファイル内のバイトオフセットは、ブロックの始めまでのファイルのオフセット (32768) にブロックのオフセット (2124) を加えたもので、ファイルのバイト位置 34892 が得られます。**

   RRDS ファイルは、レコードエディタを使用して読むことができます。あるいは `kixview` を使用して表示できます。レコードエディタについては『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を、`kixview` については『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

KSDS ファイルでのレコードの位置の特定

KSDS ファイルと代替索引ファイルは、それぞれデータファイルと索引ファイルを備えています。

データファイル	レコードを含む 1 つ以上のデータブロック。データブロック内では、すべてのレコードは主キーの順にソートされます。ただし、ブロックが順に読み込まれる場合は、1 つのブロックのレコードが、次のブロックよりも低いキーを持つとは限りません。 キーの順に各ブロックを読み込む場合、ブロックはリンクされたリストとして処理されます。ファイルの最初の物理ブロックがもっとも低いキーを含みます。 各ブロックには、次のブロック番号が含まれ、主キーの順に次のブロックを指します。キーを使用して特定のレコードを検索するには、索引ファイルを読み取る必要があります。
索引ファイル	ブロック番号と、そのブロックでもっとも高いキーで構成されるキーポインタを含みます。 索引ファイルは、データファイル内のデータブロックを指すキーポインタを含む、単一の索引ブロックで始まります。キーポインタでいっぱいになると、索引ブロックは分割されます。キーポインタの半分は新しい索引ブロックへ、残りは 2 つ目の新しいブロックに移動されます。元のブロックは、この 2 つの新しい索引ブロックを指す 2 つのキーポインタを持ちます。これにより、2 レベルの索引が作成されます。 追加の索引ブロックが必要な場合は、2 つ目の索引レベルに追加され、この索引レベルがいっぱいになるまで、データファイルを指します。2 つ目の索引レベルがいっぱいになると、3 つ目の索引レベルが作成されます。必要な数の索引レベルが作成されるまで、これが続きます。

KSDS ファイルは、レコードエディタを使用して読むことができます。または `kixview` を使用して表示できます。レコードエディタについては『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を、`kixview` については『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

VSAM データおよび索引の形式

Sun MTP ファイルの各データレコードは表 5-2 で示されている形式を持ちます。この形式は、KSDS、ESDS、および RRDS ファイルに使用されます。各レコードのデータの前には 4 バイトのレコードヘッダーがあります。レコードが VSAM ブロックサイズから 24 バイトを引いた値よりも大きい場合、レコードはスパンされ、スパンレコードの各セグメントが 4 バイトのヘッダーを持ちます。スパンレコードがあれば、常にブロックのデータ部分の先頭から始まります。スパンレコードの各セグメントは、最後のセグメントを除いて、ブロックのデータ部分全体を使用します。ブロックの残りのスペースは常に使用されません。

注 – KSDS 代替索引データファイルおよび KSDS 索引ファイルは、スパンレコードの形式を使用しません。

表 5-2 データレコードのレイアウト

オフセット	長さ (バイト)	フィールド名	フィールドの説明
0	2	Record length	後続レコードのレコード長で、4 バイトのヘッダーが含まれます。 これがスパンレコードの最初の部分である場合、このフィールドにはレコード全体の長さが入ります。スパンレコードの 2 番目以降の部分では、この値はゼロになります。RRDS ファイルの場合、この値がゼロであればレコードスロットは空です。
2	2	Spanned record length	スパンレコードにだけ使用されます。このブロックに格納されるレコードデータの大きさが入ります。
4	1 ～ 4072、 1 ～ 8168、 1 ～ 16360、 または 1 ～ 32744	Record data	実際のレコードデータ。長さは VSAM ブロックサイズによって異なります。4096 バイトのブロックサイズでレコードが 4072 バイト以下の場合、これはレコード全体になります。 レコードに 4072 バイトより多く含まれる場合、最後の部分を除いて、レコードの各部分にはちょうど 4072 バイトが含まれます。最後の部分には、レコードを完成させるために必要なバイトが含まれます。

Sun MTP の索引ファイルには、索引レベルの数によって、ほかの索引エントリを指すエントリまたはデータブロックを指すエントリが含まれます。ブロックヘッダーのブロックタイプフィールドから、特定の索引エントリがどこを指すか決定できます。

索引エントリの形式を、表 5-3 で示します。

表 5-3 索引エントリの形式

オフセット	長さ (バイト)	フィールド名	フィールドの説明
0	1 ～ 255	Record key	このキーが指すブロックの最高のキー。この規則の唯一の例外は、キーがファイルの最高のキーを含むブロックを指す場合です。この場合、キーの各文字には「0xff」が含まれます。
キーの長さ	4	Block number	データファイルまたはこのファイルの下位索引レベルのブロック番号。

第6章

プログラムのデバッグ

この章では、アプリケーションプログラムに Sun MTP とサン以外のデバッグツールを使用する際に発生する可能性のある問題について説明します。

遠隔デバッグ

遠隔デバッグを使用して COBOL プログラムをデバッグしている途中でデバッガがハングアップする場合は、次の手順を実行してデバッガを完全に終了させます。

▼ 遠隔デバッグプロセスを終了させる

1. **すべての遠隔デバッグ用ウィンドウを閉じます。**
2. `animserv` **プロセスを特定して終了させます。**
   たとえば、シェルプロンプトで、次のコマンドを入力します。

```
$ $ ps -ef | grep animserv
```

   プロセス ID (*pid*) を記録して、次のコマンドを入力します。

```
$ kill -9 pid
```

3. **遠隔デバッグを再起動するには、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。**

第7章

START によるトランザクションの非同期起動

この章では、START コマンドによって起動したトランザクションが、領域の動作に与える可能性のある影響について説明します。この章の内容は、次のとおりです。

トランザクションが起動しているように見えない

トランザクションが起動しているように見えない場合は、kixdump -A コマンドを実行します。このコマンドによって、デバッグに役立つレポートが生成されます。コード例 7-1 に示すレポート例には、データのある START、データのない START、トリガーされた START、およびバックグラウンドの START の、START コマンドで起動された 4 つの保留トランザクションが示されています。

コード例 7-1 kixdump -A 出力の例

```
05/10/2005 14:48:10 kixdump     :entering version 8.1.0 - 05/05/2005

Shared Asynchronous Start List
------------------------------

Application:<All Blanks>
KIXSYS:</net/haven/home/test/primer>
```

コード例 7-1 kixdump -A 出力の例 (続き)

```
>> Node No: 1 <<

Trans Id:SRTD             |   Trigger Time:Tue May 10 14:47:53 2005
Term  Id:<C001>           |   Started Data Len: 15
Item No:1                 |   Start Code:SD
Protect:Yes               |   Req Id (in hex):73 6F 6D 65 72 65 71 64
RTranid:ABCD              |   RTermid:<WXYZ>
RQueue:1234               |   SysId:<All Blanks>

>> Node No: 2 <<

Trans Id:ACCT             |   Trigger Time:    Tue May 10 14:47:53 2005
Term  Id:<C001>           |   Started Data Len: 0
Item No:0                 |   Start Code:S
Protect:No                |   Req Id (in hex):  34 30 30 30 31 00 00 00
RTranid:                  |   RTermid:<All Blanks>
RQueue:                   |   SysId:<All Blanks>

>> Node No: 3 <<

Trans Id:SRTD             |   Trigger Time:Tue May 10 14:47:53 2005
Term  Id:<C001>           |   Started Data Len: 0
Item No:0                 |   Start Code:QD
Protect:No                |   Req Id (in hex):  34 30 30 30 32 00 00 00
RTranid:                  |   RTermid:<All Blanks>
RQueue:                   |   SysId:<All Blanks>

>> Node No: 4 <<

Trans Id:BGTR             |   Trigger Time:Tue May 10 14:47:53 2005
Term  Id:<All Blanks>     |   Started Data Len: 0
Item No:0                 |   Start Code:S
Protect:No                |   Req Id (in hex):  34 30 30 30 32 00 00 00
RTranid:                  |   RTermid:<All Blanks>
RQueue:                   |   SysId:<All Blanks>
*** End of Asynchronous Start List - 4 entries printed ***
```

問題のトランザクションがレポートに一覧表示されている場合、端末に対して実行されているかどうかを確認します。実行されている場合、kixdump -G コマンドを実行して、端末がインストールされているか、ビジーであるかを確認します。このコマンドによって、TCT で定義されている端末の状態を示す書式付きのレポートが表示されます。端末がインストールされていないか、ビジーの場合は、これが、トランザクションが実行されなかった原因です。

トランザクションがレポートに一覧表示されていない場合、または一覧表示されているが端末に問題がない場合は、トランザクション処理プログラムがトランザクションの実行に使用できるかどうかを調べます。kixdump -u コマンドを使用して、トランザクション処理プログラムの状態に関するレポートを表示します。

トランザクション、一時記憶域、およびそれらのパフォーマンスへの影響

トランザクションが、別のトランザクションを非同期で起動するために、EXEC CICS START を実行すると、トランザクション処理プログラムは Sun MTP の間隔制御プロセッサ (unikixstrt) にメッセージを送信し、START を要求します。

unikixstrt プロセスは、保留中の START 要求のリストを管理します。START 要求は、次のような理由により保留される場合があります。

- START 要求が失効する時間に達していない。
- START によるトランザクションの実行対象となる端末が使用できない。
- トランザクションの処理に使用可能なトランザクション処理プログラムが存在しない。

unikixstrt プロセスは、各トランザクションをキューに入れ、Sun MTP のトランザクション処理プログラムがそこから作業を取り出すようにスケジュールします。キュー内のトランザクションの数が、トランザクション処理プログラムの数よりも多くなると、unikixstrt はキューに空きができるまで余分なトランザクションのスケジュールを延期します。

unikixstrt プロセスは、起動をスケジュールしているトランザクションの実行に必要なリソースがあることを確認しようとします。複数のトランザクションが START コマンドを発行している場合は、これがシステムパフォーマンスに影響を与える可能性があります。

データのある START でトランザクションが起動される場合、トランザクション処理プログラムは、データ (通常「FROM データ」と呼ばれる) を Sun MTP 一時記憶域ファイル、TEMPSTG に書き込みます。これは VSAM KSDS ファイルです。TEMPSTG に対するすべての書き込みは、キャッシュに書き込まれます。

START コマンドで起動されたトランザクション内の、データを持ち、あとに RETRIEVE が続く START は、本質的には READQ TS があとに続く WRITEQ TS です。これはデータに関するかぎり、あとに READ と DELETE の操作が続く VSAM WRITE です。キャッシュへの書き込みにもかかわらず、関連する VSAM 操作のパフォーマンスが低下する場合、次に示すような代替の方法を使用して、START コマンドで起動したトランザクションにデータを渡すことができます。

- データに対して WRITEQ TS MAIN を実行し、START コマンドの QUEUE オプション内のトランザクションに、一時記憶域のキューの名前を渡します。これによって、VSAM ディスク I/O が回避され、パフォーマンスが向上します。
- 回復可能な START トランザクションを使用します。このようなトランザクションは TEMPSTGR に書き込まれるため、キャッシュされません。
- 共通作業域 (CWA) など、別の方法を使用してデータを渡します。
- START コマンドのパラメータ、QUEUE、RTERMID、または RTRANSID を使用します。

第8章

RDBMS アプリケーションのチューニング

この章では、リレーショナルデータベース管理システム (RDBMS) アプリケーションの大きさを決定する方法について説明します。この章の内容は、次のとおりです。

RDBMS は、Sun MTP と同じくらいの大きさのメモリーを使用します。ほとんどのアプリケーションのアクティビティーには特定の実行が伴うので、このメモリーはクライアント側、つまり Sun MTP のトランザクション処理プログラム側にあります。データベースのサーバー側では、それほど大きなメモリーは要求されません。

トランザクション処理プログラムのデータセグメントには、SQL カーソルとホスト可変項目のメモリーが含まれます。実行される SQL 文の数によっては、このメモリーが増大することがあります。

Oracle RDBMS のチューニング

Oracle アプリケーションは各アプリケーションプログラムに SQL カーソル領域を割り当て、作業記憶域の標識として、割り当てに使用されるファイル名を維持します。これにより、アプリケーションは特定のプログラムの、どのカーソルが開き、どのカーソルが閉じているかを知ることができます。作業記憶域は、Sun MTP に初期化されるため、いちじるしく増大する場合があります。この状況を避けるには、`EXEC CICS` および `EXEC SQL` 文を含むオンラインプログラムをコンパイルする際に、`kixclt` に `-d oracle` オプションを付けて使用します。`kixclt` については、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

Oracle アプリケーションは、構文解析の重複を避けるために、各トランザクション処理プログラムに対して SQL カーソルキャッシュを実行します。これによりパフォーマンスは向上しますが、各トランザクション処理プログラムのメモリーも著しく増加します。通常、ユーザーはメモリーのしきい値を高く設定して、トランザクション処理プログラムのサイクルが頻繁になることを防止します。これよりも良い方法は、Oracle Net 製品 (以前の Oracle SQL*Net V 2) と Oracle のマルチスレッド化サーバーオプションを使用することです。これによって、しきい値とコア値を低減できます。マルチスレッド化サーバーオプションの使用により、SQL カーソルのメモリーはクライアント側からサーバー側に移動し、すべての SQL カーソルを中央の 1 箇所にプールできます。この解決法ではサーバーメモリーの追加が必要ですが、クライアントで必要とされるメモリーは大きく減少します。

詳細については、Oracle のマニュアルを参照してください。

Sybase RDBMS のチューニング

Sybase ソフトウェアは、アプリケーションから SQL カーソルを使用するための最適な論理を備えています。また、DB2 `BIND` プロセスに類似する、ストアドプロシージャーを使用します。ただし、Sybase ソフトウェアはストアドプロシージャーを使用した `COMMIT` および `ROLLBACK` に関する ANSI スタイルの処理をサポートしないので、アプリケーションで問題が発生する場合があります。ANSI 互換にするには、`BEGIN` トランザクションおよび `END` トランザクションを明示的に使用する必要があります。

さらに、Sybase アプリケーションは、「レコード」ロックの代わりに、「ページ」ロックを使用します。これはオンライントランザクションのパフォーマンスに影響を与える場合があります。

Sybase ソフトウェアも、クライアントとの通信に TCP/IP プロトコルを使用するので、ネットワークのトラフィック量が多くなる場合があります。ネットワークパラメータの設定については、Sybase のマニュアルを参照してください。

第9章

バッチ処理

この章では、Sun MTP でのバッチジョブの実行に関連する問題について説明します。この章の内容は、次のとおりです。

バッチ処理と回復

本来、バッチプログラムの実行には時間がかかります。回復が有効になっている場合、レコードを更新しているプログラムが SYNCPOINT を実行するまで、レコードはロックされます。このため、同じレコードを更新するオンライントランザクションを実行するユーザーは、長い間待たされることがあります。

バッチプログラムの回復を有効にする場合、プログラムが定期的に SYNCPOINTS を実行する必要があります。これにより更新がコミットされ、ほかのユーザーが、更新されたレコードにアクセスできるようになります。また、再起動可能になるようなプログラムを記述する必要があります。つまり、再起動時に、プログラムがすでにコミットされたレコードを更新しないようにする必要があります。

実行に長時間を要するバッチプログラムに対しては、回復を無効にします。回復の無効化については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

バッチ検索間隔

VCT の「Batch Search Interval」フィールドは、実行を待機しているバッチジョブを Sun MTP が問い合わせる頻度を制御します。

- 標準のバッチ環境では、$KIXBTCH ディレクトリが検索されます。
- Sun MBM 環境では、バッチノードが照会されます。

このフィールドを 1 ～ 9999 秒までの任意の秒数に設定します。デフォルトは 120 秒です。

このフィールドを低い値に設定すると、パフォーマンスが低下することがあります。

Sun MBM の操作上の問題

この節では、Sun MTP を Sun MBM とともに実行しているときに発生する可能性がある操作上の問題について説明します。

接続の失敗

領域の $KIXSYS ディレクトリをバックアップから完全に復元する必要がある場合は、サブシステムと領域の再同期も実行し、それぞれが再接続できるようにする必要があります。再同期を実行しなかった場合は、領域を起動するときに次のエラーメッセージが表示される可能性があります。

```
KIX1143E This MTP system is not configured to connect to MBM
```

▼ 領域とサブシステムを再同期させる

1. Sun MTP 領域を停止します。
2. $KIXSYS ディレクトリを復元します。
3. Sun MBM 内で BAM を起動します。
4. BAM のメインメニューで、オプション「3 Applications & Subsystems」を選択します。

5. 「Applications & Subsystems」メニューで、オプション「4 Update a Subsystem」を選択します。
6. 更新するサブシステムの数を入力します。
7. 「Update」メニューで、オプション「2 Change MTP Region」を選択します。
8. `$KIXSYS` ディレクトリのパス名を入力し、Return キーを押します。
   正しいパス名が表示されている場合は、そのまま Return キーを押します。
9. 設定画面が表示されたら、Return キーを押します。
10. BAM を終了します。
11. Sun MTP 領域を再起動します。

第10章

Sun Mainframe Administration Tool

この章では、Sun Mainframe Administration Tool (Sun MAT) と Sun Mainframe Administration Agent (Sun MAA) を起動して使用する際に発生する可能性のある問題について説明します。この章の内容は、次のとおりです。

ツールの構成と使用については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』および『Sun Mainframe Administration Tool ユーザーズガイド』を参照してください。

unikixadmin の起動の失敗

unikixadmin サーバーが正常な初期化に失敗した場合、関連するエラーメッセージがコンソールに示されます。unikixadmin サーバーは終了し、領域の遠隔管理は無効になります。

注 – 領域の起動は、通常どおり続行されます。

たとえば、unikixadmin が Sun MTP 共有メモリーへの接続に失敗すると、次のエラーメッセージがコンソールに表示されます。

```
unikixadmin:KIX0177F [unikixadmin] Cannot link to shared memory
```

これは、領域の共有メモリーの設定が、unikixadmin プロセスをサポートするように正しく構成されていないことが原因です。共有メモリー構成のガイドラインについては、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

Java 仮想マシンが構成されていない

unikixadmin サーバーでは、Java™ 仮想マシン (JVM™)[1] バージョン 1.4 以降へのアクセスが必要です。領域を起動する前に、LD_LIBRARY_PATH 環境変数を JVM ソフトウェアの位置に設定する必要があります。unikixadmin サーバーがソフトウェアの位置を特定してソフトウェアをロードすることができない場合、unikixmain.err ファイルに次のメッセージが書き込まれます。

```
ld.so.1:unikixadmin:fatal:libjvs.so:open failed:No such file or directory
03/16/2005 15:23:27 unikixmain :KIX0330E Trace dump requested:510df
03/16/2005 15:23:28 unikixmain :KIX0117F Process 123 (type a) ended
unexpectedly:exit code 0 signal 9
```

デバッグ追跡

unikixadmin サーバーが正常に起動すると、発行されたすべてのエラーメッセージは unikixmain.err ファイルに書き込まれます。ただし、診断が困難な起動の問題が発生している場合は、デバッグ情報をコンソールと unikixmain.dbg ファイルに書き込むように unikixadmin サーバーを構成する必要があります。MTPADMINTRACE 環境変数は、デバッグ追跡をオンに設定します。これにより、unikixadmin サーバーと JVM ソフトウェアに渡されるパラメータがダンプされます。

▼ デバッグ追跡を構成する

1. **MTPADMINTRACE 環境変数を、1 以上の数値に設定します。**
2. **領域を起動します。**

   unikixadmin の初期化が完了すると、unikixmain.dbg ファイルに次の形式でメッセージが書き込まれます。

   ```
   unikixadmin :registered as //:nnnn/MTP:rrrr:unikixAdmin
   ```

1. 「Java 仮想マシン」および「JVM」とは、Java プラットフォーム用の仮想マシンを意味します。

各表記の意味は次のとおりです。

nnnn は、起動時に unikixmain の -X オプションで指定されたポート番号です。

rrrr は、領域の $KIXSYS ディレクトリです。

大規模な Sun MTP 構成のチューニング

場合によっては、unikixadmin デーモンで使用するプロパティーを JVM に定義する必要があります。これは一般に、管理フレームワークで大規模な Sun MTP の構成が有効になっている場合です。大規模な構成の定義は、プログラムやマップが 5,000 個、エンドポイントが 10,000 個など、数千を超えるリソースで構成されるものです。

unikixadmin JVM のプロパティー定義

大きなオーバーヘッドをサポートするように JVM を構成する必要がある場合は、ADMIN_JVM_OPTIONS 環境変数を使用して、unikixadmin によって作成される JVM インスタンスにオプションを伝達させます。通常は java のコマンドの一部として提供される JVM のコマンド行オプションは、ADMIN_JVM_OPTIONS 環境変数に指定できます。この環境設定を、使用している領域の設定ファイルに追加します。複数のオプションを使用している場合は、必ずそれぞれを空白文字で区切ってください。各オプションは、一重引用符で囲む必要があります。

たとえば、Java オブジェクトを保持するために使用するメモリー領域のサイズを増やすには、-Xms*n* オプションと -Xmx*n* オプションを使用して、Java ヒープのサイズを定義する必要があります。Java オブジェクトプールのサイズを最大の 512M バイトに増やす必要がある場合は、ADMIN_JVM_OPTIONS 環境変数を次のように設定します。

```
export ADMIN_JVM_OPTIONS='-Xmx512m'
```

キャッシュの寿命の設定

unikixadmin デーモンは、管理下にある Sun MTP リソースを表すキャッシュを管理します。デフォルトでは、キャッシュの寿命は 1 秒です。キャッシュ寿命の期限が切れると、そのキャッシュは無効と見なされ、その後データが要求されるとキャッシュが更新されます。大規模な構成では、キャッシュの更新を実行するためのオーバーヘッドが大きくなる可能性があります。このような場合、キャッシュの寿命を長くすることをお勧めします。キャッシュの寿命は次の方法で決定します。

1. **領域内のリソースの合計数を計算します。**

   リソースとは、マップ、プログラム、端末、トランザクションなどを指します。

2. **計算した数を 5,000 の倍数に切り上げます。**

3. **リソース 5,000 個ごとに 1 秒を割り当てます。**

   たとえば、領域にプログラムとマップが 2,000 個、端末が 5,000 台存在する場合、キャッシュの寿命を 2 秒に設定します。キャッシュの寿命を設定するには、JVM の cache_lifespan プロパティを ADMIN_JVM_OPTIONS 環境変数で次のように使用します。

```
ADMIN_JVM_OPTIONS='-Dcache_lifespan=2';export ADMIN_JVM_OPTIONS
```

   JVM のコマンド行オプションについては、JVM のマニュアルを参照してください。

第11章

パフォーマンス

この章では、本番環境で Sun MTP に影響を与える要因について説明します。この章の内容は、次のとおりです。

Sun MTP 構成上の要因

この節では、アプリケーションのパフォーマンスに影響を与える Sun MTP 構成上の要因について説明します。

回復の構成

次の要因によって、回復はパフォーマンスに影響を与えます。

- 回復が「オン」か「オフ」か
- 回復ファイルのサイズ

回復が有効になっている場合は、回復で多数の入出力操作が必要になるため、使用しているアプリケーションのパフォーマンスが影響を受ける可能性があります。このオーバーヘッドのために、回復が有効である間、領域のパフォーマンスが低下することがあります。

実行に長時間を要するバッチプログラムは、大きな回復ファイルを生成する場合があります。Sun MTP では「変更前イメージ」が循環ファイルに保持されます。ファイルの終わりに達すると、変更前イメージはファイルの最初から再び始まります。プログラムまたはトランザクションが有効であるかぎり、Sun MTP では変更前イメージは上書きされません。反対に、有効な変更前イメージが上書きされそうになると、変更前イメージを書き込んだプログラムまたはトランザクションが終了します。した

がって、回復ファイルの大きさは、ある 1 つのプログラムまたはトランザクションで更新されるすべてのレコードに加えて、ほかの有効なプログラムまたはトランザクションで更新されるレコードも格納できる十分なものである必要があります。

回復とバッチ処理については、『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』を参照してください。

セマフォーと相互排他ロック

セマフォーと相互排他ロック (相互排除) のどちらを使用して実行するかにより、システムパフォーマンスに影響が生じることがあります。Sun MTP では、デフォルトで、相互排他ロックが使用されます。セマフォーを使用して実行する場合は、`unikixmain` に `-E` オプションを付けて領域を起動する必要があります。

トランザクション処理プログラム

構成するトランザクション処理プログラムの数とそれらがリサイクルする頻度により、パフォーマンスが次のような影響を受けることがあります。

- トランザクション処理プログラムのリサイクルの頻度が高すぎるあるいは低すぎるためにシステムパフォーマンスが低下する
- トランザクション処理プログラムが再起動に失敗する

`unikixmain.log` ファイルをチェックし、このような問題の解決に役立つ、終了と再起動に関するメッセージがないかどうか調査してください。

また、開発システムにアクセスする個々のシステムトランザクション、たとえば `CMNU` や `CTBL` が、トランザクション処理プログラムを制御する点にも注意してください。これらのトランザクションを終了するまで、トランザクション処理プログラムは解放されません。

複数のトランザクション処理プログラムの影響

トランザクション処理プログラムの数は VCT に定義され、領域で実行可能なバックグラウンドタスクの最大数、バッチジョブの最大数、およびデバッグ端末の数が指定されます。構成のガイドラインについては、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

一般に、バックグラウンドタスクの数は、トランザクション処理プログラムの数の 50% を超えてはいけません。ただし、アプリケーションが `START` によるバックグラウンドトランザクションの起動 (一時データを `TRIGGER` でトリガーまたは `START` で起動したトランザクション) に依存することが多い場合は、トランザクション処理プ

ログラムの設定値の 75% の値を設定できます。バックグラウンドタスクに設定した値が高すぎる場合、大半のトランザクション処理プログラムがバックグラウンドタスクを実行するため、オンラインユーザーはパフォーマンスの低下を経験します。

バッチジョブの総数は、トランザクション処理プログラムの数の 50% を超えてはいけません。設定した値が高すぎる場合、大半のトランザクション処理プログラムがバッチジョブを実行するため、オンラインユーザーはパフォーマンスの低下を経験します。

バックグラウンドタスクの最大数とバッチジョブの最大数の合計は、トランザクション処理プログラムの数以下にする必要があります。そうでない場合、エラーメッセージが表示され、値を変更するまで VCT を保存できません。

VCT で指定されたデバッグ端末の数がトランザクション処理プログラムの数の合計よりも大きい場合、大半のトランザクション処理プログラムがデバッグタスクを実行することになるため、オンラインユーザーはパフォーマンスの低下を経験することがあります。

リサイクルの頻度

トランザクション処理プログラムが終了して再起動するのは正常です。ただし、トランザクション処理プログラムがあまりにも頻繁にリサイクルする場合は、領域のメモリーしきい値と最大コアの値が正しく構成されていないことが考えられます。この場合は、次のエラーメッセージがログファイルに書き込まれる可能性があります。

```
KIX0299I Process memory (%x) exceeds threshold (%x) - terminating txn
processor
```

`-M c` の値は増やさずに、`-M t` の値を増やしてみてください。これらのオプションについては、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

リサイクルの失敗

`CINI` トランザクションまたは `CEMT...NEWCOPY` トランザクションを実行してトランザクション処理プログラムをリサイクルしたときに、一部のトランザクション処理プログラムがリサイクルされない場合、次のようなメッセージが `unikixmain.log` ファイルに書き込まれることがあります。

```
03/16/2005 09:11:25 unikixtran53:kxcobcall: call cobcancel ppt = <0> local = <59>
03/16/2005 09:11:25 unikixtran53:kxcobcall: call cobcancel ppt_plad = <0> local =
<e9a4560>
03/16/2005 09:11:25 unikixtran53:kxsvnstcobws: program was cancelled, do nothing
```

ログファイルにこのようなメッセージが多数ある場合は、トランザクション処理プログラムがソケット接続を待機していたか、会話型トランザクションが完了していないなどの理由で、リサイクルが行われていない可能性があります。このような問題のあるトランザクションが実行中でないことを確認してから、CINI トランザクションまたは CEMT...NEWCOPY トランザクションを使用してリサイクルを実行してください。

共有メモリー

領域で共有メモリーを獲得できない場合、次のようなメッセージがログファイルに書き込まれます。

```
0147F Shared memory not available (%d bytes requested)
```

kixdump -s validate コマンドでも、領域が追加の共有メモリーをどのように獲得したかの詳細が提供されます。

次の手順は、領域が共有メモリーを使い果たしたときの対処法を説明しています。

▼ 共有メモリーの割り当てを増やす

1. **アプリケーションを調べて、どれだけの共有メモリーが割り当てられているかを確認します。**

   次の 4 つを探します。

   - タイプ MAIN のWRITEQ TS
   - -TM オプションを持つ unikixmain
   - GETMAIN SHARED 文
   - 領域内で有効な多数の COBOL プログラム

2. **unikixmain に -S オプションを付けて領域を再起動し、これまでよりも大きな値を指定して、共有メモリーの割り当てを増やします。**

   たとえば、Solaris のプラットフォームでは、16M バイトが起動時のデフォルト値です。共有メモリーを 24M バイトに増やすには、次のように入力します。

   ```
   $ unikixmain -S 24M other-options
   ```

3. **共有メモリーの割り当てを増やしても問題が解決されない場合は、kixsnap ユーティリティーを使用してシステムのスナップショットを取り込み、ご購入先まで送付してください。**

   kixsnap については、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

Sun MTP でローカルプロセスメモリーおよび共有メモリーを使用する方法と、領域で共有メモリーを構成するためのガイドラインの詳細は、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

VSAM 構成上の要因

この節では、パフォーマンスに影響を与える可能性がある VSAM の構成上の要因について説明します。

Sun MTP ファイルの使用状況の特定

Sun MTP の各トランザクション処理プログラムは、アプリケーションに必要なファイルと内部で使用するファイルを開きます。どのアプリケーションファイルを開くかは、ファイル管理テーブル (FCT)、宛先管理テーブル (DCT)、およびジャーナル管理テーブル (JCT) によって決定されます。これらのテーブルについては、『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。

各Sun MTP アプリケーションファイルには 1 つ以上の物理ファイルが必要です。これらのファイルの一覧を次の表に示します

表 11-1 Sun MTP 物理ファイル

タイプ		物理ファイルの数
VSAM ファイルタイプ	キーシーケンス (KSDS)	1 + ファイルのセグメント数 + 2 * 代替キーの数
	相対レコード (RRDS)	1
	入力順 (ESDS)	1
DCT エントリタイプ	パーティション内 (宛先タイプ = `FILE`)	2
	パーティション内 (宛先タイプ = `TERMINAL`)	1
	パーティション外	1
	遠隔パーティション	1
JCT エントリ	すべてのエントリ	1
システムごとの追加ファイル (デフォルトで定義)	VSAM カタログ	2
	一時記憶域	2

各トランザクション、回復、およびメインサーバーのプロセスは、表 11-1 に示すすべてのファイルを開きます。オペレーティングシステムは、各サーバープロセスが開くことができるファイルの最大数を制限します。この調整可能なカーネルパラメータが NOFILES です。単一のトランザクション、回復、またはメインサーバーに割り当てるファイルの数がシステムの制限を超える場合、ファイルをすべて開こうとしたときにサーバー障害が発生します。

もしほかの Sun MTP サーバーまたはプロセスが同じシステムに常駐する場合、これらのサーバーまたはプロセスに必要なファイルの数を、開くファイルの必要数に加える必要があります。

ブロックサイズ

ブロックサイズは、アプリケーションのパフォーマンスに影響を与える場合がありますが、Sun MTP ソフトウェアは特定のブロックサイズを持つファイルシステムを必要としません。ただし、最適なパフォーマンスを得るには、VSAM ファイルは Sun MTP と同じブロックサイズのファイルシステムに存在する必要があります。

unikixmain に -b オプションを付けると、システム全体の VSAM ブロックサイズが K バイト (KB) 単位で指定されます。可能な値は、4、8、16、または 32 です。デフォルトのブロックサイズは 4K バイトです。このオプションは、VSAM カタログがない場合だけ適用されます。カタログが存在する場合、VSAM ブロックサイズは自動的にカタログのブロックサイズに等しくなります。ブロックサイズの変更については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

注 – ファイルのブロックサイズは、VSAM カタログのブロックサイズと同じである必要があります。そうでなければ、ファイルを開けません。

一時記憶域

Sun MTP のデフォルト動作では、一時記憶域キュー (TSQ) の情報はディスク (補助記憶装置) に書き込まれます。この動作は、領域の起動時に、unikixmain に -TM オプションを使用することで変更できます。引数 M は、Sun MTP が TSQ 情報をメモリー (主記憶装置) に書き込むように命令します。WRITEQ TS コマンドに MAIN オプションを使用する方法でも、デフォルトの動作を変更できます。

メモリー管理の変更

Sun MTP のこれまでのリリースでは、項目が TSQ に書き込まれると、Sun MTP はその項目に使用するメモリーを共有メモリーのプールから割り当てました。キューが削除されると、メモリーはプールに戻されました。TSQ 項目のそれぞれについてメモリーを割り当てたり、割り当てを解除したりしていると、膨大な量のメモリー管理のオーバーヘッドが発生する可能性があります。

領域の起動時に TSQ ブロックサイズを指定することにより、メモリー管理のオーバーヘッドを低減できます。これにより、キューが作成されて最初の項目が書き込まれるときに、指定したサイズのブロックが Sun MTP によって割り当てられます。そのあとの書き込みでは、Sun MTP はこのブロックから項目の空き領域を割り当てるだけです。最初の書き込み以降、共有メモリーのオーバーヘッドは発生しません。ブロックがいっぱいになると、キューの全項目を保持するために必要なブロックが追加で割り当てられます (共有メモリーの割り当てが必要です)。キューが削除されると、そのキューに対して割り当てられていたブロックは「空き」リストに置かれ、ほかの TSQ で使用できるようになります。「空き」リストを実装することにより、空き領域を共有メモリーに戻すオーバーヘッドが不要になります。既存のキュー、または今後作成されるキューは、空きリストのメモリーブロックを使用できます。初期に割り当てられるメモリーブロックが空きリストで見つかれば、メモリーブロックを初期に割り当てる必要もなくなる場合もあります。詳細は、59 ページの「空きリストからのブロックの再割り当て」を参照してください。

ブロックサイズ

Sun MTP でサポートされる TSQ ブロックのサイズは、512 バイト、1K バイト、2K バイト、4K バイト、8K バイト、16K バイト、32K バイトです。起動時に `unikixmain` に `-TM` オプションを指定すると、TSQ データのデフォルトのブロックサイズは 512 バイトになります。このデフォルトサイズは、起動オプション `-q` を指定することでも変更きできます。

`-q` オプションに有効な値は、`-q1` (1K バイトブロック)、`-q2` (2K バイトブロック)、`-q4` (4K バイトブロック)、`-q8` (8K バイトブロック)、`-q16` (16K バイトブロック)、`-q32` (32K バイトブロック) です。

起動オプションと WRITEQ TS API との相互作用

次の表に、各起動オプションと、MAIN または AUXILIARY のオプションを付けた WRITEQ TS コマンドの組み合わせを使用した結果を示します。

表 11-2 起動オプションと WRITEQ TS コマンドの使用

-T オプション	-q オプション	WRITEQ TS コマンド	結果
使用しない	使用しない	MAIN オプションも AUXILIARY オプションも使用しない	すべての TSQ 項目がディスクに書き込まれます。
使用しない	使用しない	MAIN オプション	WRITEQ TS...MAIN コマンドに指定された TSQ 項目が、デフォルトである 512 バイトのブロックサイズまたは項目の保持に十分な大きさのサイズを使用して、メモリー内のキューに書き込まれます。その他の TSQ 項目はディスクに書き込まれます。
使用しない	使用する	MAIN オプション	WRITEQ TS...MAIN コマンドに指定された TSQ 項目が、-q オプションに指定されたブロックサイズまたは項目の保持に十分な大きさのサイズを使用して、メモリー内のキューに書き込まれます。その他の TSQ 項目はディスクに書き込まれます。
使用しない	使用しない	AUXILIARY オプション	WRITEQ TS...AUXILIARY コマンドに指定された TSQ 項目がディスクに書き込まれます。
引数 M	使用しない	MAIN オプションも AUXILIARY オプションも使用しない	すべての TSQ 項目が、デフォルトである 512 バイトのブロックサイズまたは項目の保持に十分な大きさのサイズを使用して、メモリー内のキューに書き込まれます。
引数 M	使用する	MAIN オプションも AUXILIARY オプションも使用しない	すべての TSQ 項目が、-q オプションに指定されたブロックサイズまたは項目の保持に十分な大きさのサイズを使用して、メモリー内のキューに書き込まれます。
引数 M	使用する	MAIN オプション	すべての TSQ 項目が、-q オプションに指定されたブロックサイズまたは項目の保持に十分な大きさのサイズを使用して、メモリー内のキューに書き込まれます。
引数 M	使用しない	AUXILIARY オプション	WRITEQ TS...AUXILIARY コマンドに指定された TSQ 項目がメモリーに書き込まれます。
引数 A	使用しない	MAIN オプション	すべての TSQ 項目がディスクに書き込まれます。

unikixmain については、『Sun Mainframe Transaction Processing ソフトウェアリファレンスマニュアル』を参照してください。

ブロックと空き領域の割り当て

キューに割り当てられている既存のブロック内に空き領域がない場合、および空きリストに空きブロックがない場合、Sun MTP は TSQ 項目に対してメモリーの新しいブロックを割り当てる必要があります。領域で構成されたブロックサイズよりもレコードのサイズが大きい場合は、保存されるレコードを保持するために十分な大きさのサイズが割り当てられます。

Sun MTP では、複数のメモリーブロックに項目が分割されることはありません。そのため、100 バイトの項目が書き込まれる場合、キューに割り当てられているメモリーブロックの中に 100 バイトの空きがない場合は、その項目を収めるための新しいブロックが割り当てられます。キューに書き込まれる項目のサイズと種類によっては、メモリーブロック内の仮想アドレス空間が使用できないこともあります。たとえば、512 バイトのブロックサイズが使用されていて、項目のサイズが 100 バイトの場合、100 バイトの項目を 5 つ、つまり 500 バイトをブロックに書き込むことができます。これにより、1 ～ 12 バイトの項目を書き込まないかぎり、このブロックには使用できない 12 バイトが残ります。

空きリストからのブロックの再割り当て

Sun MTP では、異なるブロックサイズごとに別個の空きリストが管理されます。各ブロックは、次の方法で空きリストから割り当てられます。

- 書き込まれる新しい項目が、領域に構成済みのブロックサイズよりも小さい場合、使用可能なブロックを求めて構成済みのブロックサイズの空きリストが検索されます。
- 新しい項目のサイズが、領域に構成済みのブロックサイズよりも大きい場合、該当するブロックサイズの空きリストが検索されます。すべてのブロックサイズが検索されるわけではありません。これにより、大きなブロックサイズを、それが作成される原因となったと思われる大きな項目に使用することができます。

使用するブロックサイズの決定

Sun MTP では、最大 32K バイトまでの任意のサイズのキュー項目が適切に処理されます。アプリケーションが 32K バイトよりも大きなキュー項目を書き込もうとすると、エラーになります。

ブロックサイズを選択する前に、アプリケーションが一時記憶域キューを使用する方法を分析する必要があります。キューの数と合計サイズを考慮して、必要な空き領域の合計量とキューの最適なサイズを決定してください。

次の各例は、アプリケーション環境に応じて特定のブロックサイズが適切になる理由を説明しています。

- アプリケーションが、通常は合計 32K バイトを超えるデータを各 TSQ に書き込む場合は、32K バイトのブロックサイズが最適です。共有メモリーブロックの割り当てが最小限になり、パフォーマンスが最大になります。
- アプリケーションが、3,000 ～ 4,000 バイトのデータをほとんどの TSQ に書き込む場合は、4K バイトよりも大きいブロックサイズを選ぶよりも、4K バイトのブロックサイズを選ぶ方が適切です。このような場合に 4K よりも大きいブロックサイズを使用すると、仮想アドレス空間が浪費されたり、使われなかったりする可能性があります。1 つ以上のキューに対して 4K バイトを超える書き込みが時折発生する場合、Sun MTP はそのキューに 4K バイトよりも大きなブロックを割り当て、さらに大きな項目が書き込まれる場合はさらに大きなブロックを割り当てます。
- アプリケーションが、通常は 256 バイトの TSQ 項目を書き込む場合は、`-q` 起動オプションを使用して、ブロックサイズを 1K バイトに設定することを検討してください。このサイズを設定すると、アプリケーションはブロックごとに 4 つのキュー項目を格納できるため、メモリーを効率的に使用できます。アプリケーションが、4,000 バイトなど通常よりもかなり大きな項目を時折書き込む場合、Sun MTP はその大きな項目を保持するために、自動的に 4K バイトのブロックを割り当てます。多数の項目がキューに書き込まれる場合は、ブロックサイズを大きくした方が効率的です。

一時記憶域キューの状態の表示

`kixdump -q` コマンドを使用すると、有効な一時記憶域キューに関する情報が表示されます。情報には、キューの名前、キューのタイプ (補助または主)、キューの合計サイズ、キューから最後に読み取られた項目、キューに最後に書き込まれた項目、有効な項目の数、最初に使用可能な項目番号があります。キューのタイプが主の場合は、そのキューに割り当てられているデータブロックに関する情報も追加して表示されます。これには、データブロックのアドレス、キューの次のデータブロックを指すポインタ、ブロック内の使用可能領域または空き領域を指すポインタ、ブロック内で使用可能なバイト数、ブロック内の項目の最大サイズ (バイト単位)、ブロックに収められている項目のカウント、書き換えアクティビティー後のブロック内の未使用領域の合計 (バイト単位)、ブロックサイズ (バイト単位) があります。

カーネル構成

使用している実行環境が相互排他ロックではなくセマフォーを使用するように設定されている場合は、ユーザーまたはシステム管理者が `/etc/system` ファイル内のセマフォーの値を調べて、それが十分であることを確認する必要があります。表 11-3 に、関係のあるセマフォーと正しい値の設定例を示します。

表 11-3　セマフォーの値

セマフォーの名前	説明	領域の要件	Sun MBM ノードの要件	/etc/system ファイルでの値の設定例
SEMMNI	Solaris システム内のセマフォーセットの最大数。	領域ごとに最大 3 セット	ノードごとに 4 セット	領域が 2 つある場合、値は (2*3) = 6 Sun MBM ノードも 1 つ存在する場合は 4 を加算 (6+4 = 10)
SEMMNS	システム内のセマフォーの最大数。	領域ごとに最大 51 個のセマフォー	ノードごとに 5 個のセマフォーにアクティビティー (スレッド) の数を加算	領域が 2 つある場合、値は (2*35) = 70 Sun MBM ノードも 1 つ存在し、アクティビティーが 10 個ある場合は、15 を加算 (70+15 = 85)
SEMMSL	セマフォーセットごとのセマフォーの最大数。	MTP では現在、1 つのセットで最大 40 個のセマフォーを使用。	n/a	40 以上の値を設定

その他の TSQ 情報

Sun MTP は、`WRITEQ TS` コマンドにはじめて TSQ が指定されたときに、TSQ を動的に作成します。TSQ は一時記憶域テーブル (TST) であらかじめ定義することもできます。これらのキューはローカルでも遠隔でもかまいません。TST 内に定義できる TSQ の数に制限はありません。各 TSQ は 32,768 個のレコードを収めることができ、各レコードは最大 32,768 バイトのデータを収めることができます。

主一時記憶域は補助記憶域よりも多くのメモリーを必要とするため、寿命が短いか頻繁にアクセスされる小さなキューに主に使用します。使用しているアプリケーションが大量のデータを TSQ に書き込み、そのデータの寿命が比較的長いかデータがほとんどアクセスされない場合は、補助一時記憶域の使用を検討してください。

補助記憶域に書き込まれた TSQ は回復可能として定義できますが、主記憶域内のキューではできません。

I/O ボトルネック

アプリケーションを実行している際に、I/O ボトルネックの結果としてパフォーマンスの低下に直面することがあります。VSAM ファイルが極めて大きい場合または Sun MTP アカウンティング用に選択したジャーナルのバッファーサイズが小さすぎる場合に、I/O ボトルネックが発生します。

大きい VSAM ファイル

VSAM ファイルが極めて大きい場合、I/O ボトルネックが発生します。Sun MTP のスパンファイル機能を使用して、大きいファイルを複数のファイルシステムに割り当てることができます。KSDS ファイルごとに最大 8 つのセグメントを指定できます。

Sun MTP は起動時に構成されたブロックサイズを使用して、データを書き込みます。結果として、トランザクションからレコードへのアクセスが単一のデバイスのスループットに制限されなくなり、スループットが改善します。

小さすぎるジャーナルバッファーサイズ

Sun MTP アカウンティング用にジャーナルを使用する場合は、32K バイトのバッファーサイズを使用します。これによってアカウンティングレコード (それぞれが約 0.5K バイト) はメモリー上にバッファーリングされ、バッファーが 32K バイトの大きさになった時点でディスクにフラッシュされます。32K バイトのバッファーサイズは、ファイルの I/O を減少させ、パフォーマンスを向上させます。

注 – バッファーサイズが大きい場合、メモリーに格納されるアカウンティングデータの量が増えます。その結果、システムクラッシュの際に消失するアカウンティングデータの量が増加します。Sun MTP アカウンティングジャーナルの最小バッファーサイズは、1024 バイトです。

VSAM ファイルの低いスループット

アプリケーションが、回復なしとして構成されている VSAM ファイルを使用する場合は、VSAM キャッシュ書き込み機能を使用して I/O スループットを改善できます。この機能は、VSAM ファイルへの物理的書き込みをシステムキャッシュに保持することによって、Sun MTP システムが VSAM ファイルへの物理的書き込みを延期することを可能にします。キャッシュ機能の使用については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

VSAM データセットのバッファーの不足

多数の VSAM データセットを使用し、システムメッセージ「`No unused buffers`」が周期的に表示される場合は、多数の異なるファイルを使用する有効なトランザクションが多数存在する可能性があります。次のいずれかの方法を実行して、この問題を解決します。

- バッチジョブを実行している場合は、オンラインアクティビティーが最小となる時間帯にバッチジョブをスケジュールし直すことによって、バッファーの競合を取り除くことができる場合もあります。

- VSAM データセットに定義されているブロックサイズを超える大きいデータレコードがある場合は、ブロックサイズを増大させます。デフォルトの Sun MTP ブロックサイズは 4,096 (4K バイト) ですが、8K、16K、または 32K バイトのブロックサイズに増大できます。

  ブロックサイズを変換する場合は、VSAM カタログをエクスポートしてから、インポートする必要があります。詳細については、『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』を参照してください。

- Sun MTP では共有バッファーを使用して VSAM ファイルとの読み書きを行うため、共有バッファーの数を増やし、それらが割り当てられるバッファープールの数を増やすことを検討した方がよい場合があります。

  デフォルトで、Sun MTP はバッファープールの半分を VSAM 索引ブロックのために予約します。残りの半分は、VSAM データブロックのために使用されます。また、Sun MTP はデフォルトでバッファーの 50% を索引ブロックに、50% をデータブロックに割り当てます。

  VSAM バッファーの割合は、`unikixmain` に `-I` オプションを使用して変更できます。このパラメータは、VSAM 索引ブロックのために予約する VSAM バッファーの割合を、25 ～ 75 の間の値で指定します。

使用しているアプリケーションが、VSAM バッファープールの数を増やす候補として適切かどうかを決定するには、Sun Mainframe Administration Tool (Sun MAT) または `kixdump` ユーティリティーを使用します。

▼ Sun MAT を使用して VSAM バッファープールの使用方法を決定する

1. **パフォーマンスに問題があると思われるアプリケーションを起動します。**
2. **Sun MAT を起動します。**

   『Sun Mainframe Transaction Processing ソフトウェア 管理者ガイド』および『Sun Mainframe Administration Tool ユーザーズガイド』を参照してください。

3. **分析しようとするアプリケーションが動作中の領域の領域ウィンドウを開きます。**
4. **「Advanced」タブをクリックします。**
5. **ドロップダウンメニューで「System Gates」を選択します。**

   1 つの `KXSEMBUF` ゲートが 1 つのバッファープールを表しています。

6. **ウィンドウに「Total Waits」列、「Maximum Waiting」列、「Total Wait Time」列が表示されていることを確認します。**

7. **各 KXSEMBUF ゲートについて「Total Waits」列、「Maximum Waiting」列、「Total Wait Time columns」列を調べます。**

   「Total Waits」列は、領域の起動後にこのゲートが所有された回数を示します。この値は、領域の起動時に 0 に初期化され、ゲートがロックされるたびに増分されます。

   「Maximum Waiting」列は、リクエスタがこのゲートの待機を要求した最大数を示します。

   「Total Wait Time」列は、プロセスがこのゲートを待機した合計回数を示します。この値は、ゲートに対する競合がどのくらいあるかを示します。

8. **これらの列のいずれかで 1 つ以上のゲートが高い値を示す場合は、バッファープールの数を増やし、それに応じて領域のバッファーの数を増やすことを検討してください。バッファープール 1 つにつきバッファー 8 個で計算します。**

▼ kixdump を使用して VSAM バッファープールの使用方法を決定する

1. **パフォーマンスに問題があると思われるアプリケーションを起動します。**

2. **kixdump -u コマンドを約 15 秒ごとに実行して「Waiting for」という表現を検索し、リソースが使用可能になる状態を多数の Sun MTP サーバープロセスが待機しているかどうかを判断します。**

```
$ kixdump -u |grep “Waiting for”
```

   - 「Waiting for semaphore」という表現が見つかった場合は、Sun MTP のロックの 1 つを待機中であることを示します。セマフォービットマスクはロックを表します。KXSEMBUF ロックは、VSAM バッファープールに対応するロックです。50% を超えるなど大量のサーバープロセスが KXSEMBUF を待機している場合、このアプリケーションはバッファープールを増やす最良の候補です。
   - 「Waiting for locked record」という表現が見つかった場合は、VSAM レコードキーの待機を表します。このタイプの待機は、アプリケーションに関連するものであることが普通で、バッファープールの数を増やしても効果はありません。アプリケーションのプログラマに相談して、このタイプの待機が必要かどうかを判断してください。
   - 「Waiting for file buffer」という表現が見つかった場合は、Sun MTP の VSAM バッファーの 1 つを待機中であることを示します。このタイプのロックの競合は、バッファーに含まれるレコードの数を減らすことで解決できる場合があります。

     VSAM バッファープールの数および共有バッファーの数の構成については、『Sun Mainframe Transaction Processing ソフトウェア 構成ガイド』を参照してください。

プログラム上の要因

プログラムの設計方法がパフォーマンスに影響することがあります。

会話型トランザクション

アプリケーションに会話型トランザクションがあると、パフォーマンスの問題が発生することがあります。会話型トランザクションでは、トランザクションが有効な間、ユーザーとの会話 (通常、SEND/RECEIVE のシーケンス) が行われます。これは、トランザクション中にユーザーとの会話がない擬似会話型トランザクションと異なります。通常、擬似会話型トランザクションは、SEND、次に RETURN TRANSID で終わります。ユーザーは応答を入力しますが、トランザクションは有効ではありません。ユーザーが Enter キーまたは PF キーを押すと、TRANSID で指定されるトランザクションの新規インスタンスが始まります。

Sun MTP 回復機能を使用する場合は、会話型トランザクションに重大な設計上の影響があります。回復機能がオンになっていると、トランザクションは、トランザクションの終わりまたは同期点まで、変更した VSAM レコードに対する制御を保持します。これは、ユーザーからの応答を待っている間、会話型トランザクションが何秒または何分にもわたって、レコードの制御を保持することがあるということを意味します。レコードがすでに書き込まれている場合にも、これは当てはまります。回復機能のないシステムでは、この時点で、ほかのトランザクションがレコードを使用できるようになります。

設計上のもう 1 つの問題は、トランザクションが完了する前にユーザーからの応答が必要な場合、トランザクションの最大時間に際限がなくなるという事実から生じます。ユーザーには更新が完了したように見えますが、実際には完了していません。トランザクションが完了するまで、この更新はデータベースにコミットされません。何分も経過したあとで、トランザクションが失敗した場合、ユーザーは更新がロールバックされたことに気付かないことがあります。

会話型トランザクションを完全に回避するか、すでに会話型になっているシステムの場合はレコードの保持がありうる位置に存在するすべての RECEIVE コマンドの直前に SYNCPOINT コマンドを追加することによって、これらの問題を防止できます。

過度のスワッピングまたはページング

過度のスワッピングまたはページングは、システムパフォーマンスを低下させます。オペレーティングシステムでスワッピングまたはページングが過度に起きる場合は、トランザクション処理プログラムの数とメモリーのしきい値を減少させる必要がある場合があります。アプリケーションにこれらの値が必要な場合、システムの実メモリーを増やさなければならない可能性があります。

C 言語のプログラムと仮想記憶

使用しているアプリケーションで C プログラムが広範に使用され、仮想記憶が不足し始めている場合は、領域の設定ファイルの `KIX_PGMTXN_MODE` 環境変数の値を `TXSERIES` に設定できます。この設定により、トランザクションが終了すると、C 言語アプリケーションのすべての共有オブジェクトが閉じます。

詳細は『Sun Mainframe Transaction Processing ソフトウェア 開発者ガイド』の C の共有オブジェクトの使用方法に関する章を参照してください。

第 12 章

診断ツール

この章では、Sun MTP が提供する追跡機能とダンプ機能について説明します。ご購入先から指示を受けないかぎり、これらの機能を使用しないでください。この章では、次のトピックについて説明します。

Sun MTP 追跡機能の使用

Sun MTP の追跡機能を使用して、選択的に下位レベルのルーチンを追跡できます。この機能は、ご購入先の技術者の指導のもとでのみ使用します。この内部の追跡機能によって、Sun MTP が割り当てる共有メモリーの量が増加します。次の各プロセスには、共有メモリー内で追跡テーブルが割り当てられています。

- `unikixmain`
- `unikixprt`
- `unikixstrt`
- `unikixept`
- `unikixrcv`
- `unikixsched`
- `unikixtran`# (各 `unikixtran` は固有の追跡テーブルを持つ)

各追跡テーブルのデフォルトサイズは、80,000 バイトです。

次の公式は、Sun MTP 領域の追跡にどれだけの共有メモリーが必要かを示します。

(トランザクション処理プログラムの数 + トランザクションクラスの数 + 5) * 80,000 = **バイト**

トランザクション処理プログラムとトランザクションクラス (存在する場合) の数を 5 に加算し、公式を適用します。次の例では、領域に 8 つのトランザクション処理プログラムと 4 つのトランザクションクラスがあります。したがって、17 個のプロセスを使用して、共有メモリーの要件が求められます。

(8 + 4 + 5) * 80,000 = 1,360,000 バイト

Sun MTP は、内部の追跡に 1,360,000 バイトの共有メモリーを割り当てます。

拡張追跡機能の共有メモリー要件を計算してから、その数をアプリケーションの共有メモリー要件に加えて、共有メモリー要件の合計を算出します。Sun MTPの起動時に、`unikixmain` の `-S` オプションにこの値を指定します。

追跡管理ユーティリティー

`kixetrace` コマンドを使用して、「Trace Administration Utility」メニューを表示します。

注 – 追跡管理ユーティリティーは、ご購入先の指示があった場合のみ使用してください。

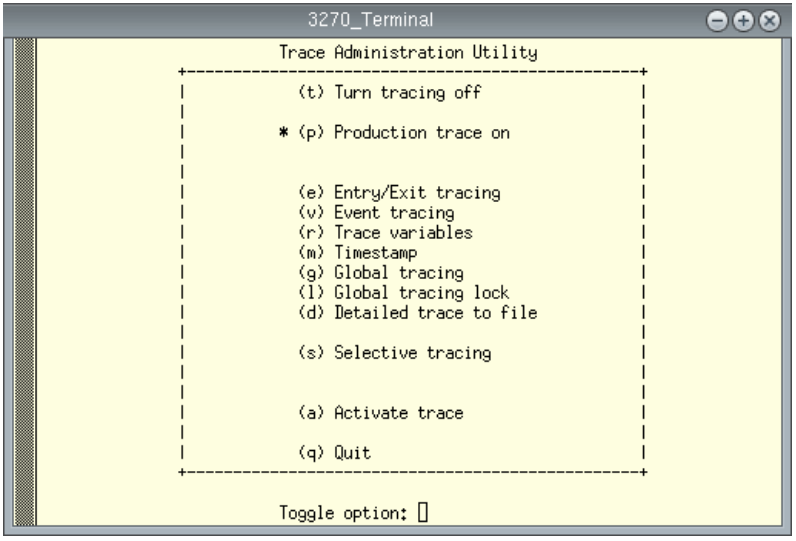

図 12-1 「Trace Administration Utility」メニュー

選択されたオプションの横にアスタリスク (*) が表示されます。

追跡機能を最初に実行したときは、常に「Production trace on」が選択されて表示されます。これは、本番追跡がオンになっているという意味ではなく、デフォルト表示です。メニューは、Sun MTP で実行されているオプションを必ずしも示していません。

ダンプ機能

Sun MTP には、問題の原因の判定に使用できるシステム情報をダンプする方法が 2 つあります。

- `kixsnap` ユーティリティー。詳細は『Sun Mainframe Transaction Processing ソフトウェア リファレンスマニュアル』を参照してください。
- アプリケーションダンプ機能

アプリケーションダンプ機能

アプリケーションダンプ機能は、次の場合に、アプリケーションの実行環境を一覧するための書式付きダンプファイルを生成します。

- 実行時エラーが発生
- アプリケーションプログラムが `HANDLE ABEND` コマンドなしで、CICS `ABEND` コマンドを使用
- 記憶域がない状態などのエラーが発生

アプリケーションダンプファイルは、`$KIXSYS` ディレクトリに書き込まれます。ファイル名は、メッセージ番号 KIX6702I とともにログファイルに出力されます。ダンプファイル名は、トランザクション名とシーケンス番号の組み合わせで、たとえば、`AC010000.prt` のようになります。

書式付きダンプファイルには、次の情報が含まれます。

- エラーの日時
- 実行時エラー情報メッセージ
- CICS コマンド追跡情報。「tombstone」とも呼ばれます
- ヒープのアドレス、共有メモリー、最大コア、およびメモリーしきい値
- Exec interface block (EIB)
- 共通作業域 (CWA) および端末ユーザー領域 (TCTUA)
- 逆順にしたプログラムスタック
- 通信領域 (`COMMAREA`)
- COBOL プログラムの作業記憶域セクション

また、アプリケーションダンプ機能は、次の情報を unikixmain.err と unikixmain.log に書き込みます。

- 実行時エラーメッセージ (ある場合)
- CICS コマンド追跡情報 (tombstone)
- ダンプファイルの名前

ダンプ機能の有効化と無効化

アプリケーションダンプファイルはデフォルトで生成されますが、次のいずれかのオプションを使用して、ダンプ機能を明示的に有効にできます。

- unikixmain -D dy
- kixcontrol -d y

ダンプ機能を無効にするには、次のどちらかのオプションを使用します。

- unikixmain -D dn
- kixcontrol -d n

EXEC CICS GETMAIN 呼び出しを使用して取得した共有メモリーのダンプを有効にするには、KIXGETMAINDMP 環境変数を設定する必要があります。たとえば、領域の Korn シェルの設定ファイルに変数を設定するには、次のように入力します。

```
KIXGETMAINDMP=1;export KIXGETMAINDMP
```

unikixmain.err と unikixmain.log のエントリ

アプリケーションダンプ機能が有効になっている場合、エントリは、unikixmain.err と unikixmain.log に書き込まれ、次の例のように表示されます。

```
05/10/2005 13:40:40 unikixtran2 :-------------------------------------------
05/10/2005 13:40:40 unikixtran2 :Transaction AC01  program ACCT01    has abended
08/19/2002 13:40:40 unikixtran2 :Function ENDBR             CID# 00001113
Condition FILENOTFOUND
Condition FILENOTFOUND
05/10/2005 13:40:40 unikixtran2 :Transaction was  executing MTP code
05/10/2005 13:40:40 unikixtran2 :Previous functions:
05/10/2005 13:40:40 unikixtran2 :Function READNEXT          CID# 00001100
Condition Successful

05/10/2005 13:40:40 unikixtran2 :Function READNEXT          CID# 00001100
Condition Successful
05/10/2005 13:40:40 unikixtran2 :Function READNEXT          CID# 00001100
Condition Successful
05/10/2005 13:40:40 unikixtran2 :-----------------------------------------------
05/10/2005 13:40:40 unikixtran2 :Formatted dump requested, file is AC010000.prt
.
.
. (more log not shown)
```

この例では、CID# 00001113 が失敗したコマンドです。ソースプログラム ACCT01.cbl では、0001113 の検索によって次のコードが表示されます。

```
* EXEC CICS ENDBR DATASET('ACCTIT') END-EXEC.
       MOVE 'ACCTIT' TO DFHEIV1
       MOVE 0 TO DFHEIV11
       MOVE '&2                    $   #00001113' TO DFHEIV0
       CALL 'kxdfhei1' USING DFHEIV0,
           DFHEIV1,
           DFHEIV11,
           DFHEIV99
       GO TO 9999-DFHEXIT DEPENDING ON DFHEIGDK.
```

ダンプファイルの出力

COBOL プログラムに対して生成されるダンプファイルは、コード例 12-1 の例のようになります。

ダンプファイルの情報を解釈する場合は、次の点に注意してください。

- 作業記憶域は、スタック内のプログラムごとに表示されます。
- `EXEC CICS LINK` 呼び出しを使用してリンクされているプログラムには、`COMMAREA` が表示されます。
- `GETMAIN SHARED` セグメントは、`EXEC CICS GETMAIN SHARED` 呼び出しを使用して取得されたメモリーの領域全体のリストを表します。
- `GETMAIN LOCAL` セグメントは、`EXEC CICS GETMAIN` 呼び出しを使用してトランザクションプロセスが取得したローカルメモリーのリストを表します。
- EIB のフィールドは、16 進数の値を返します。

コード例 12-1　ダンプファイル —COBOL (1 / 3)

```
05/10/2005  11:21:32.000  unikixtran0 Formatted Dump File

EXEC CICS ABEND called with ABCODE(NEST)

Transaction:NST9 has abended in Program:NESTABC
Failure occurred in Sun MTP code
CICS Command Level Trace:
      Function ABEND            CID# 00000086 Incomplete
      Function LINK             CID# 00000091 Condition Successful
      Function GETMAIN          CID# 00000090 Condition Successful
      Function GETMAIN          CID# 00000085 Condition Successful

Current address on heap     (36dd48)
Address of Shared Memory    (ee000000)
Address of Max Core         (ee000000)
Address of Threshold Memory (edf80000)
```

コード例 12-1　　ダンプファイル —COBOL (2 / 3)

```
******************  Exec Interface Block   ******************
EIBTIME   0112131c              EIBSYNC   00
EIBDATE   0103302c              EIBFREE   00
EIBTRNID  NST9                  EIBRECV   00
EIBTASKN  0000002c              EIBSEND   00
EIBTRMID  C000                  EIBATT    00
DFHEIGDI  0000                  EIBEOC    00
EIBCPOSN  0004                  EIBFMH    00
EIBCALEN  0009                  EIBCOMPL  00
EIBAID    27                    EIBSIG    00
EIBFN     0e0c                  EIBCONF   00
EIBRCODE  000000000000          EIBERR    00
EIBDS                           EIBERRCD  00000000
EIBREQID                        EIBSYNRB  20
EIBRSRCE                        EIBNODAT  00
EIBRESP   00000000
EIBRESP2  00000000

*********************************   CWA   *********************************
No Common Work Area present

*********************************   TCTUA   *******************************
TCTUA LENGTH = 255
ee201ee0 00000000  20202020 20202020 20202020 20202020  *                *
ee201ef0 00000010  20202020 20202020 20202020 20202020  *                *
ee201f00 00000020  20202020 20202020 20202020 20202020  *                *
ee201f10 00000030  20202020 20202020 20202020 20202020  *                *
ee201f20 00000040  20202020 20202020 20202020 20202020  *                *
ee201f30 00000050  20202020 20202020 20202020 20202020  *                *
ee201f40 00000060  20202020 20202020 20202020 20202020  *                *
ee201f50 00000070  20202020 20202020 20202020 20202020  *                *
ee201f60 00000080  20202020 20202020 20202020 20202020  *                *
ee201f70 00000090  20202020 20202020 20202020 20202020  *                *
ee201f80 000000a0  20202020 20202020 20202020 20202020  *                *
ee201f90 000000b0  20202020 20202020 20202020 20202020  *                *
ee201fa0 000000c0  20202020 20202020 20202020 20202020  *                *
ee201fb0 000000d0  20202020 20202020 20202020 20202020  *                *
ee201fc0 000000e0  20202020 20202020 20202020 20202020  *                *
ee201fd0 000000f0  20202020 20202020 20202020 202020    *                *

*********************   GETMAIN SHARED MEMORY   **************************

LENGTH=10
ee1ff7c0 00000000  73737373 73737373 7373              *ssssssssss      *
```

コード例 12-1　　ダンプファイル —COBOL (3 / 3)

```
*********************   GETMAIN SHARED MEMORY   *************************

LENGTH=10
ee1ff760 00000000  73737373 73737373 7373              *ssssssssss      *

*********************   GETMAIN LOCAL MEMORY   **************************

LENGTH=10
3656c8 00000000  6C6C6C6C 6C6C6C6C 6C6C                *llllllllll      *

*********************   GETMAIN LOCAL MEMORY   **************************

LENGTH=10
364ae0 00000000  6C6C6C6C 6C6C6C6C 6C6C                *llllllllll      *

*****************************  PROGRAM STACK ************************

PROGRAM:NESTABC
WORKING STORAGE
36bcf0 00000000  20202020 20202020                     *                *

PROGRAM:NEST2L
WORKING STORAGE
36b390 00000000  20202020 20202020 4E455354 324C2020   *        NEST2L  *
36b3a0 00000010  20202020 20202020                     *                *

COMMAREA
COMMAREA LENGTH: 9
368e38 00000000  4E455354 354C2020 20                  *NEST5L          *

PROGRAM:NEST5L
WORKING STORAGE
3669f8 00000000  20202020 20202020 0000000A 20202020   *        ....    *

COMMAREA
COMMAREA LENGTH: 9
365cd0 00000000  4E455354 394C2020 20                  *NEST9L          *

PROGRAM:NEST9L
WORKING STORAGE
3669f8 00000000  20202020 20202020 0000000A 20202020   *        ....    *

COMMAREA
No commarea present
```

C プログラムに対して生成されるダンプファイルは、コード例 12-2 の例のようになります。

コード例 12-2　　ダンプファイル —C (1 / 2)

```
05/10/2005  15:40:09.000  unikixtran2 Formatted Dump File

Run-Time Error:
C Application ERROR

Transaction:XTBM has abended in Program:XXTBMU
Failure occured in Application code
CICS Command Level Trace:
      Function GETMAIN          CID# 00000020 Condition Successful
      Function SEND MAP         CID# 00000013 Condition Successful
      Function ADDRESS          CID# 00000045 Condition Successful
      Function ADDRESS          CID# 00000001 Condition Successful

Current address on heap     (400b1000)
Address of Shared Memory    (4086c000)
Address of Max Core         (4086c000)
Address of Threshold Memory (4056c000)

******************  Exec Interface Block   ******************
EIBTIME   0154008c              EIBSYNC   00
EIBDATE   0099067c              EIBFREE   00
EIBTRNID  XTBM                  EIBRECV   00
EIBTASKN  0000002c              EIBSEND   00
EIBTRMID  C003                  EIBATT    00
DFHEIGDI  0000                  EIBEOC    00
EIBCPOSN  0004                  EIBFMH    00
EIBCALEN  000c                  EIBCOMPL  00
EIBAID    27                    EIBSIG    00
EIBFN     0c02                  EIBCONF   00
EIBRCODE  000000000000          EIBERR    00
EIBDS                           EIBERRCD  00000000
EIBREQID                        EIBSYNRB  20
EIBRSRCE                        EIBNODAT  00
EIBRESP   00000000
EIBRESP2  00000000

********************************   CWA   ***************************
No Common Work Area present
```

コード例 12-2　　ダンプファイル —C (2 / 2)

```
********************************   TCTUA   ************************
TCTUA LENGTH = 255
c24a0a80 00000000  20202020 20202020 20202020 20202020  *                *
c24a0a90 00000010  20202020 20202020 20202020 20202020  *                *
c24a0aa0 00000020  20202020 20202020 20202020 20202020  *                *
c24a0ab0 00000030  20202020 20202020 20202020 20202020  *                *
c24a0ac0 00000040  20202020 20202020 20202020 20202020  *                *
c24a0ad0 00000050  20202020 20202020 20202020 20202020  *                *
c24a0ae0 00000060  20202020 20202020 20202020 20202020  *                *
c24a0af0 00000070  20202020 20202020 20202020 20202020  *                *
c24a0b00 00000080  20202020 20202020 20202020 20202020  *                *
c24a0b10 00000090  20202020 20202020 20202020 20202020  *                *
c24a0b20 000000a0  20202020 20202020 20202020 20202020  *                *
c24a0b30 000000b0  20202020 20202020 20202020 20202020  *                *
c24a0b40 000000c0  20202020 20202020 20202020 20202020  *                *
c24a0b50 000000d0  20202020 20202020 20202020 20202020  *                *
c24a0b60 000000e0  20202020 20202020 20202020 20202020  *                *
c24a0b70 000000f0  20202020 20202020 20202020 202020    *                *

********************************   COMMAREA   *************************
COMMAREA LENGTH: 12
c2335880 00000000  44656C74 61585442 4D540000           *DeltaXTBMT..    *

*****************************  WORKING-STORAGE *************************
PROGRAM:XXTBMU
7afcd3e0 00000000  00                                    *.               *
```

トランザクションが不正終了した位置の特定

unikixmain.log ファイルには、トランザクションの失敗についての情報が収められています。トランザクション名、プログラム名、トランザクションで実行された最後の 4 つの CICS コマンドが提供されます。各 CICS コマンドは、次の情報を提供します。

- 実行された CICS コマンドの名前
- CID# 参照。CICS コマンドの ID 参照番号を表します
- 実行されたコマンドの状態

CID# 参照は、実行された CICS コマンドの位置を特定するために役立ちます。この参照番号は、kixclt トランスレータによって生成されます。kixclt トランスレータの出力 (COBOL の場合 *.cbl ファイル) で CID# 参照を検索して、実行された CICS コマンドの位置を特定できます。

用語集

A

ABEND　(名詞) タスクを異常終了させるための EXEC CICS コマンドのオプション。

C

COMMAREA　(名詞) 通信領域。任意の端末と通信する複数のタスク間で、データの受け渡しに使用される領域です。この領域を使用して、タスク内のプログラム間でデータを受け渡しすることもできます。

CWA　(名詞) 共通作業域。システム内の任意のタスクからアクセスできる必要があるユーザーデータを保持するために、アプリケーションプログラムで使用できるタスク共有プール内の領域。この領域は、システムの初期化中に取得されます。

E

EBCDIC　(名詞) 拡張 2 進化 10 進コード。多くのデータ処理システム、データ通信システム、および関連装置で情報交換に使用される、8 ビット符号化文字から構成された符号化文字セット。

exec interface block (EIB)
: (名詞) CICS プログラム内の各タスクに関連する制御ブロック。EIB には、アプリケーションプログラムの実行中に役立つ情報 (トランザクション ID など) と、プログラムのデバッグのためにダンプを使用する際に役立つ情報が収められます。

M

`make` 機能
: (名詞) Sun MTP システムの再構築に使用されるユーティリティー。

S

SQL
: (名詞) 構造化照会言語。一連の情報へのアクセスと更新に使用されるリレーショナルデータベース言語。

Sun Mainframe Batch Manager ソフトウェア (Sun MBM)
: (名詞) 管理された環境でバッチジョブを実行するための機能を提供するバッチマネージャー製品。Sun MBM は、バッチの本番負荷を処理し、開始時刻、バッチプロセスの最大数、ジョブの優先順位などの割り当てられたパラメータによってジョブをスケジューリングします。

Sun Mainframe Transaction Processing ソフトウェア (Sun MTP)
: (名詞) プロセス間通信サービス、ソケット、および COBOL をはじめとする複数言語などの機能を使用してアプリケーションを実行するユーザーアプリケーション。クライアント以外の Sun MTP のすべてのコンポーネントは、メインサーバープロセスである `unikixmain` によって起動します。

Sun MTP 領域
: (名詞) システム上の異なるアプリケーションを定義するプロセス、リソース、および環境変数のセット。

T

Table Manager
: (名詞) Sun MTP テーブルで領域のリソースの定義に使用する Sun MTP の機能。

TCP/IP (名詞) インターネットの基礎となるネットワークプロトコル群。伝送制御プロトコル (TCP) は、信頼性の高い全二重データストリームを提供するプロトコルです。インターネットプロトコル (IP) は、TCP のパケット配信サービスを提供するプロトコルです。TCP プロトコルは、ユーザープロセスではなく、IP と連携します。

TCTUA (名詞) 端末ユーザー領域。端末に関連するトランザクション間で、データの受け渡しに使用されます。

TN3270 サーバー (名詞) TCP/IP - TN3270 プロトコルを使用して、PC、Macintosh、および UNIX システムで実行される 3270 エミュレータを Sun MTP でサポートできるようにします。TN3270E もサポートします。

TN3270 プロトコル (名詞) 従来の TCP/IP Telnet プロトコルの拡張機能。IBM-3270 のような ASCII 以外のブロックモードデバイス、および Sun MTP などのアプリケーションが TCP/IP を介して通信できるようになります。TN3270E も含まれています。

U

`unikixmain` サーバー (名詞) Sun MTP メインサーバープロセス。

`unikixrc.cfg` ファイル (名詞) `unikixdcl` サーバー、`unikixqm` サーバー、および `unikixtnemux` サーバーについての情報を収めたリソースファイル。起動時に、Sun MTP 通信マネージャーは、`unikixrc.cfg` ファイルを読み取り、適切なサーバーを起動します。

V

VSAM クラスタ (名詞) VSAM 構造での最上位の命名。クラスタの名前は主アクセス名になります。クラスタには、データ定義、索引定義、および任意の二次索引も含まれます。

VSAM 構成テーブル (VCT) (名詞) 基本の Sun MTP 構成パラメータを定義する制御テーブル。

VSAM データセット (名詞) VSAM 規則に従って編成、格納、およびアクセスされる関連データの名前付きコレクション。

い

一時記憶域テーブル (TST)
(名詞) ローカルおよび遠隔の一時記憶域キューの記憶域と回復を定義する Sun MTP テーブル。

か

会話型トランザクション
(名詞) トランザクションが有効な間、ユーザーとの会話 (通常は SEND/RECEIVE シーケンス) が行われるトランザクション。

仮想記憶アクセス方式 (VSAM)
(名詞) さまざまなアクセス方式によってレコードにアクセスする方式。次のようなものがあります。

ESDS (入力順データセット)。レコードは順次に記録され、アクセスされます。

RRDS (相対レコードデータセット)。レコードは、データセット内で占める位置番号によって検索されます。

KSDS (キーシーケンスデータセット)。レコードは索引またはキーによって検索されます。

環境変数
(名詞) プログラムファイルおよびアプリケーションの位置を定義する変数。環境変数は、クライアントとサーバーの両方で使用されます。

き

キーシーケンスデータセット (KSDS)
(名詞) キーによって参照される可変長レコードの索引編成 VSAM ファイル。

こ

顧客情報管理システム (CICS) (名詞) ユーザー作成のアプリケーションプログラムによって、遠隔端末で入力されたトランザクションの並行処理を可能にする IBM の使用許諾を受けたプログラム。データベースの構築、使用、および保守のための機能が含まれています。

さ

索引ファイル (名詞) ブロック番号とそのブロックでの最上位のキーで構成されるキーポインタが収容されています。キーはレコードを指します。

し

システムネットワーク体系 (SNA) (名詞) 情報単位を伝達し、ネットワークの構成と動作を制御するための論理構造、形式、プロトコル、および操作順序。

ジョブ制御言語 (JCL) (名詞) オペレーティングシステムに対するジョブを特定し、ジョブの要件を記述するために使用される制御言語。

す

スパンファイル (名詞) 複数のファイルシステムに渡ってセグメント化されるファイル。

せ

セグメント (名詞) スパンファイルの一部分。スパンファイルを参照。

そ

相対レコードデータセット (RRDS) (名詞) データセット内で占める位置番号でレコードが検索される VSAM データセット。

た

端末管理テーブル (TCT) (名詞) 端末、プリンタ、および遠隔システム接続の特定情報を収めた Sun MTP テーブル。

つ

追跡機能 (名詞) `kixdump` コマンドを使用してアクセスされるメモリーに追跡エントリを作成します。デバッグに使用します。

て

データセグメント (名詞) クラスタのデータ部分を収容するファイル。

データセット VSAM データセットを参照。

データファイル (名詞) 内部にレコードを収容する 1 つまたは複数のデータブロックで構成されます。

と

同期点 (名詞) アプリケーションプログラムの実行における論理点。この論理点で行われたプログラムによるデータベースの変更は、一貫性があり完全で、データベースにコミットできます。出力はこの点まで持続されてから宛先に送信され、入力はメッセージキューから削除され、データベースの更新はほかのアプリケーションで利用できるようになります。プログラムが異常終了した場合、回復機能と再起動機能によって、更新は前回完了した同期点よりも前にはバックアウトしません。

に

入力順データセット (ESDS) (名詞) データレコードを入力順に収容する可変長 VSAM ファイル。

は

パーティション外キュー (名詞) 「DCT –Extrapartition Destinations」画面で特定されるキューに書き込まれるすべてのデータを収めた順編成ファイル。このファイルは、エントリで指定したレコード形式と長さで開きます。

ふ

ファイル管理テーブル (FCT) (名詞) Sun MTP アプリケーションプログラムがアクセスする VSAM データファイルについての情報を収めた Sun MTP テーブル。各ファイルには一連の特性が関連付けられています。この特性を Sun MTP のコマンドルーチンで使用することにより、アプリケーションプログラムが指定したコマンドが検証され実行されます。

不正終了 (名詞) タスクの異常な終了。アプリケーションは、`EXEC CICS ABEND` コマンドを実行してタスクを異常終了させることができます。「異常終了」と同じ意味です。

プログラム管理テーブル (PCT) (名詞) Sun MTP でトランザクションの特定と初期化に使用する制御情報を収めた Sun MTP テーブル。

プログラムリストテーブル (PLT) (名詞) システムの起動時、ユーザーの起動時、またはシステムの停止時に Sun MTP によって自動的に開始されるプログラム名を収めた Sun MTP テーブル。

ま

マニュアルページ (名詞) `man` コマンドを使用して、コマンドの使用方法を表示できます。たとえば、`grep` コマンドについて表示するときは、プロンプトで `man grep` と入力します。

り

領域 Sun MTP 領域を参照。

索引

L

M

N

O

R

S

T

U

V

W

X

あ

い

え

か

き

く

こ

さ

し

す

せ

そ