

# Tmax プログラミングガイド (RCA)

Tmax v6.0



Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

## Copyright Notice

Copyright © 2016 TmaxSoft Co., Ltd. All Rights Reserved.

45, Jeongjail-ro, Bundang-gu, Seongnam-si, Gyeonggi-do, 13613, South Korea

## Restricted Rights Legend

All TmaxSoft Software (Tmax®) and documents are protected by copyright laws and international convention. TmaxSoft software and documents are made available under the terms of the TmaxSoft License Agreement and this document may only be distributed or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TmaxSoft Co., Ltd.

Nothing in this software document and agreement constitutes a transfer of intellectual property rights regardless of whether or not such rights are registered) or any rights to TmaxSoft trademarks, logos, or any other brand features. This document is for information purposes only. The company assumes no direct or indirect responsibilities for the contents of this document, and does not guarantee that the information contained in this document satisfies certain legal or commercial conditions. The information contained in this document is subject to change without prior notice due to product upgrades or updates. The company assumes no liability for any errors in this document.

このソフトウェア(Tmax®)マニュアルの内容とプログラムは、日本国の著作権法および国際条約によって保護されています。マニュアルの内容とプログラムは、TmaxSoft Co., Ltd.との使用許諾契約書の下でのみ使用することができ、マニュアルは使用許諾契約で許可されている範囲を除いては、配布または複製することができません。TmaxSoftの書面による事前の承諾を得ることなく、このマニュアルの全部または一部を電子的または機械的な方法を問わず、転送、複製、配布したり、または二次的著作物を作成する等の行為を一切禁じます。

このソフトウェアのマニュアルとプログラムの使用許諾契約は、いかなる場合においても、マニュアル及びプログラムと関連する知的財産権(登録の有無を問わず)を譲渡するものと解釈されず、TmaxSoftのブランド、ロゴ、商標等の使用権限を与えるものではありません。マニュアルは、情報を提供する目的でのみ提供しており、これに伴う契約上の直接的ないしは間接的な責任を負わず、マニュアルの内容は法律上もしくは商業的な特定の条件が満たされることを保証しません。マニュアルの内容は、製品のアップグレード及び修正により、その内容が予告なく変更されることがあり、内容上の誤りがないことを保証しません。

## Trademarks

Tmax®, Tmax WebtoB® and JEUS® are registered trademark of TmaxSoft Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.

Tmax®, Tmax WebtoB®, JEUS® は、TmaxSoft Co., Ltd.の登録商標です。その他、記載されている会社名、製品名などは、各社の商標または登録商標です。

## Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : openssl-0.9.7.m, zlib-1.1.4, expat-2.0.0, net-snmp, DCE1.0, pthread, google-diff-match-patch, libevent, getopt

---

Detailed Information related to the license can be found in the following directory :  
\${INSTALL\_PATH}/license/oss\_licenses

この製品の一部ファイルまたはモジュールは、openssl-0.9.7.m、zlib-1.1.4、expat-2.0.0、net-snmp, DCE1.0、pthread、google-diff-match-patch、libevent、getoptライセンスを遵守します。

詳細情報については、製品ディレクトリーの\${INSTALL\_PATH}/license/oss\_licensesに記載されている事項を参照してください。

## 文書情報

文書名: Tmax プログラミングガイド (RCA)

発行日: 2016年8月5日

ソフトウェアバージョン: Tmax v6.0

ガイドバージョン: v2.1.1

---



# 目次

このガイドについて .....	ix
<b>第1章 紹介 .....</b>	<b>1</b>
1.1. 概要 .....	1
1.2. 構成 .....	1
1.3. システム構造 .....	2
1.3.1. サーバー・モード .....	2
1.3.2. クライアント・モード .....	3
1.4. 特徴 .....	3
<b>第2章 使用と管理 .....</b>	<b>5</b>
2.1. 環境設定 .....	5
2.2. システムの使用 .....	6
2.2.1. ローカル・モード .....	7
2.2.2. リモート・モード .....	8
2.3. システム管理 .....	9
2.3.1. 起動および終了 .....	9
2.3.2. 監視 .....	10
2.4. 障害対策 .....	11
2.4.1. rcakill .....	12
<b>第3章 APIとエラーメッセージ .....</b>	<b>13</b>
3.1. API .....	13
3.1.1. tpsetfd .....	13
3.1.2. tpclrfd .....	16
3.1.3. tpissetfd .....	18
3.1.4. tpschedule .....	21
3.1.5. tpuschedule .....	22
3.1.6. tpreMOTEconnect .....	24
3.1.7. tpgetrcAhseqno .....	24
3.1.8. tpgetrcainfo .....	24
3.2. エラーメッセージ .....	25
<b>第4章 RCAHの例 .....</b>	<b>27</b>
4.1. RCAHカスタマイズ .....	27
4.2. 例 .....	27
4.2.1. Makefile .....	27
4.2.2. プログラム .....	28
<b>索引 .....</b>	<b>33</b>



## 図目次

[図 1.1]	サーバー・モードでのプロセス・フロー .....	3
[図 1.2]	クライアント・モードでのプロセス・フロー .....	3





# このガイドについて

## 対象読者

本書は、Tmax<sup>®</sup>(以下、Tmax) RCAを用いて開発を行う開発者を対象としています。

RCAはRaw Client Agentの略語です。Tmaxクライアント・ライブラリーを使用できない従来の通信プログラムやPDAなどをTCP/IPソケットを介してTmaxシステムと接続し、Tmaxシステムが提供するサービスを使用できるようにサポートするエージェントです。同書では、RCAモジュールのインストールおよび環境設定方法について説明します。

## 前提知識

本書は、Tmaxシステムの概要とTmaxシステムが提供する各種機能や特性などを習得するための手引書です。

本書を理解するためには、以下の事項についての知識が必要です。

- ミドルウェアおよびUNIXシステムについての基本知識
- Tmaxの基本概念
- JavaとCプログラミングについての知識

## 制限事項

本書を読む前にTmaxの基本概念を熟知している必要があります。実務上の具体的な使用方法や管理・運用についての内容は、各製品ガイドを参照してください。

---

### 参考

Tmaxシステムの開発についての基本的な内容は、『Tmax 運用ガイド』および『Tmax アプリケーション開発ガイド』を、Tmaxが提供するコマンドとC APIについては、『Tmax リファレンスガイド』を参照してください。

---

# 本書の構成

本書は、計4章で構成されています。

各章の主な内容は以下のとおりです。

- 第1章: 紹介

RCAシステムの概要およびシステム構造について説明します。

- 第2章: 使用と管理

RCAシステムの使用方法や管理方法について説明します。

- 第3章: APIとエラーメッセージ

RCAで使用するAPI関数の使用方法とエラーメッセージについて説明します。

- 第4章: RCAHの例

RCAHカスタマイズと例について説明します。

## 表記上の規則

表記	意味
<AaBbCc123>	プログラム・ソースコードのファイル名、ディレクトリー
<Ctrl> + C	CtrlキーとCキーを同時に押す
[Button]	GUIのボタン、メニュー名
太字	強調
「」、『』（鍵カッコ）	関連文書、あるいはガイド内の他の章および節の表示
「入力項目」	画面UI上の入力項目
ハイパーリンク	メール・アカウント、Webサイト
>	メニューの実行順
+ ----	下位ディレクトリー/ファイル有り
----	下位ディレクトリー/ファイル無し
<div>参考</div>	参照/注意事項
[図1.1]	図の名前
[表1.1]	表の名前
AaBbCc123	コマンド、コマンド実行結果の画面出力、サンプル・コード
[ ]	オプション・パラメータ値
	選択パラメータ値

## システム要件

	要求事項
プラットフォーム	IBM AIX 5.x / 6.1 / 7.1
	HP-UX 11.xx
	SunOS 5.7~5.9 / SunOS 5.10 / SunOS 5.11
ハードウェア	1GB以上のハードディスク空き容量
	512MB以上のメモリー空き容量
データベース	Oracle 9~12
	Tibero 4~5
	DB2
	Informix

## 関連文書

ガイド	説明
Tmax 運用ガイド	Tmaxを利用するための環境設定ファイルとシステム運用方法について説明しています
Tmax アプリケーション開発ガイド	Tmaxアプリケーション・プログラムの開発で使用するAPIの概念と使用方法および例について説明しています
Tmax リファレンスガイド	Tmaxアプリケーションの開発に使用するコマンドおよびクライアントとサーバーの接続、通信に使用する関数の使用方法と例について説明しています
Tmax メッセージリファレンスガイド	Tmax製品の使用時に発生する可能性のあるメッセージ(エラー・メッセージを含む)と、その対応方法について説明しています

## お問合せ先

### Korea

TmaxSoft Co., Ltd.  
45, Jeongjail-ro, Bundang-gu,  
Seongnam-si, Gyeonggi-do, 13613  
South Korea  
Tel: +82-31-8018-1000  
Fax: +82-31-8018-1115  
Email: [info@tmax.co.kr](mailto:info@tmax.co.kr)  
Web (Korean): <http://www.tmaxsoft.com>  
TechNet: <http://technet.tmaxsoft.com>

### USA

TmaxSoft Inc.  
101 North Wacker Drive, Suite 2014,  
Chicago, IL 60606  
U.S.A  
Tel: +1-312-525-8330  
Email: [info@tmaxsoft.com](mailto:info@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/us\\_en/home](http://www.tmaxsoft.com/us_en/home)

### Japan

TmaxSoft Japan Co., Ltd.  
5F Sanko Bldg, 3-12-16 Mita, Minato-Ku, Tokyo, 108-0073  
Japan  
Tel: +81-3-5765-2550  
Fax: +81-3-5765-2567  
Email: [info@tmaxsoft.co.jp](mailto:info@tmaxsoft.co.jp)  
Web (Japanese): <http://www.tmaxsoft.co.jp>

## China

Beijing TmaxSoft System Software Co., Ltd.  
Room103, No.2 Huizhong Building, Seven Street Shangdi,  
Haidian District, Beijing, 100085  
P.R.China  
Tel: +86-10-6298-8827  
Email: [info@tmaxsoft.com.cn](mailto:info@tmaxsoft.com.cn)  
Web (Chinese): [http://www.tmaxsoft.com/cn\\_en/home\\_cn\\_en](http://www.tmaxsoft.com/cn_en/home_cn_en)

## Brazil

Tmax Brasil Sistemas e Serviços Ltda.  
Av. Copacabana, 177, sala 32~35 Empresarial 18 do Fortel  
Alphaville Barueri, Sao Paulo, 06472-001  
Brazil  
Tel: +55-11-4191-3100  
Fax: +55(11) 4191-3705 (extension#112)  
Email: [info.bra@tmaxsoft.com](mailto:info.bra@tmaxsoft.com)  
Web (Portuguese): [http://www.tmaxsoft.com/br\\_en/home\\_br\\_en](http://www.tmaxsoft.com/br_en/home_br_en)

## Russia

Tmax Rus L.L.C.  
Leninsky prospekt, 113/1 (Park Place Moscow),  
Office 318e, Moscow, 117198  
Russia  
Tel: +7(495)970-01-35  
Email: [info.rus@tmaxsoft.com](mailto:info.rus@tmaxsoft.com)  
Web (Russian): [http://www.tmaxsoft.com/ru\\_ru/home\\_ru\\_ru](http://www.tmaxsoft.com/ru_ru/home_ru_ru)

## Singapore

Tmax Singapore Pte. Ltd.  
430 Lorong 6, Toa Payoh #10-02,  
OrangeTee Building, 319402  
Singapore  
Tel: +65-6259-7223  
Fax: +65-6258-7112  
Email: [info.sg@tmaxsoft.com](mailto:info.sg@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/sg\\_en/home\\_sg\\_en](http://www.tmaxsoft.com/sg_en/home_sg_en)

## United Kingdom

TmaxSoft UK Ltd.  
215 Knyvett House, Watermans Business Park,  
The Causeway, Staines TW18 3BAB  
United Kingdom  
Tel: +44-1784-895005  
Email: [info.uk@tmaxsoft.com](mailto:info.uk@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/gb\\_en/home\\_gb\\_en](http://www.tmaxsoft.com/gb_en/home_gb_en)

## Canada

TmaxSoft Canada, Inc.  
2425 Matheson Blvd East, 8th floor,  
Unit 824 Mississauga, ON, L4W 5K4  
Canada  
Tel: +1-905-361-2888  
Email: [info.canada@tmaxsoft.com](mailto:info.canada@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/ca\\_en/home\\_ca\\_en](http://www.tmaxsoft.com/ca_en/home_ca_en)



## Australia

TmaxSoft Proprietary Limited  
L32, 101 Miller Street, North Sydney 2060  
Australia  
Tel: +91-9845-330-704  
Email: [info.aus@tmaxsoft.com](mailto:info.aus@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/au\\_en/home\\_au\\_en](http://www.tmaxsoft.com/au_en/home_au_en)

## India

TmaxSoft Technologies Private Limited  
Sobha Alexander Plaza, 3rd Floor,  
16/2 Commissariat Road, Bangalore-560025  
India  
Tel: +91-9845-330-704  
Email: [info.india@tmaxsoft.com](mailto:info.india@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/in\\_en/home\\_in\\_en](http://www.tmaxsoft.com/in_en/home_in_en)

## Turkey

TmaxSoft Co., Ltd. Turkey Liaison Office  
Windowist Tower. Eski Buyukdere Cad. No:26,  
Maslak 34467 Istanbul  
Turkey  
Tel: +90-544-553-6045  
Email: [cslee@tmaxsoft.com](mailto:cslee@tmaxsoft.com)  
Web (English): [http://www.tmaxsoft.com/tr\\_en/home\\_tr\\_en](http://www.tmaxsoft.com/tr_en/home_tr_en)



# 第1章 紹介

本章では、RCAシステムの構成、基本構造、特徴について説明します。

## 1.1. 概要

RCA(Raw Client Agent)は、Tmaxクライアント・ライブラリーを使用できない従来の通信プログラムとTCP/IPソケットで接続し、Tmaxシステムで提供するサービスを利用できるようにサポートします。

RCAはマルチスレッド(Multi Thread)方式で処理され、それぞれのスレッドはTmaxクライアントに該当します。したがって、起動されるスレッドの数はTmaxシステムのライセンスを考慮して設定する必要があります。またRCAではPOSIXスレッドが使用され、カーネルレベルのスレッドでシステムによってスケジューリングされます。したがって、マルチCPU環境でより効果的に処理されます。

RCAは、構成する場所に応じてローカル・モードとリモート・モードに分けられます。

- ローカル・モード(Local Mode)

Tmaxシステム内に位置するように構成して、サーバーまたはクライアントの動作を処理する方式です。

- リモート・モード(Remote Mode)

Tmaxシステム外部のノードに位置させ、サーバーまたはクライアントの動作を処理する方式です。

各モードについての詳細内容は、[「2.2. システムの使用」](#)を、システム構造についての詳細内容は、[「1.3. システム構造」](#)を参照してください。

## 1.2. 構成

RCAは以下のように構成されます。

- RCAL

RCALは、TCP/IP構造でサーバープロセスに該当します。RCALは相手通信プログラムからの接続を待ち、接続情報をRCAHに渡す役割をします。RACLは相手通信プログラムには接続しません。

以下のパスに存在しており、ユーザーの接続を制御します。

```
$TMAXDIR/bin/rcal
```

- RCAH

ユーザーのロジックと一緒に作成されます。

RCAHは、Tmaxクライアント・スレッド・ライブラリー(\$TAMXDIR/lib/librcah.so)とユーザーが作成したクライアント・プログラムが結合して作成される実行ファイルであり、ユーザーのロジックと一緒に作成されます。RCAHはマルチスレッド方式で処理され、1つのスレッドが1つのTmaxクライアントにあたります。つまり、1つのクライアント・プログラム(RCAH)がスレッド数分のTmaxシステムと接続を確立してサービスを提供します。

RCALは最大500のRCAHを起動することができ、1つのRCAHは最大60のスレッドを持つことができます。したがって、1つのRCAモジュールは最大30000個(500\*60)のクライアントをサポートします。このようにマルチスレッド方式で処理されるため、グローバル変数または静的変数の使用に注意を払う必要があります。RCAHプログラムの例は、「[第4章 RCAHの例](#)」を参照してください。

- 管理ツール

RCAは、rcastatとrcakillを使ってRCA情報を監視および終了することができます。

管理ツール	説明
rcastat	RCA設定情報と現在RCAに接続しているクライアントの数を監視することができます。 rcastatについての詳しい内容は、「 <a href="#">2.3.2. 監視</a> 」を参照してください
rcakill	RCAを終了する際、RCAが使用したリソースを削除するために使用します

## 1.3. システム構造

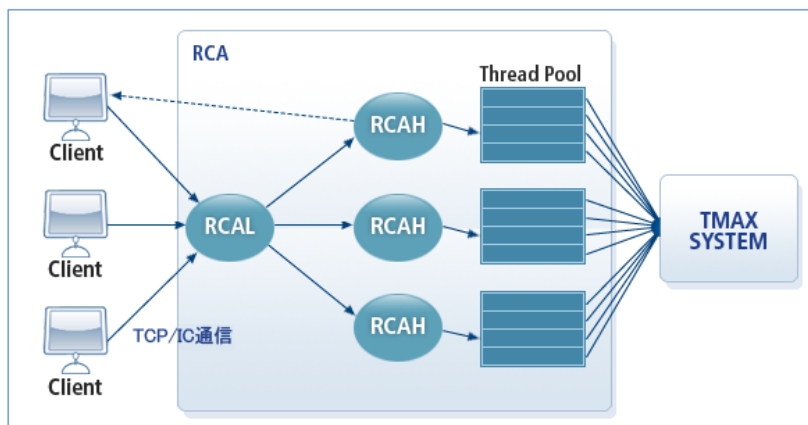
システムは、クライアント要求の処理に応じてサーバー・モードとクライアント・モードに分けられます。

### 1.3.1. サーバー・モード

サーバー・モードは、RCALがクライアントのリモート接続に待機してから処理する方式です。

以下の図は、サーバー・モードでのRCALとRCAHのプロセス・フローです。

【図 1.1】 サーバー・モードでのプロセス・フロー



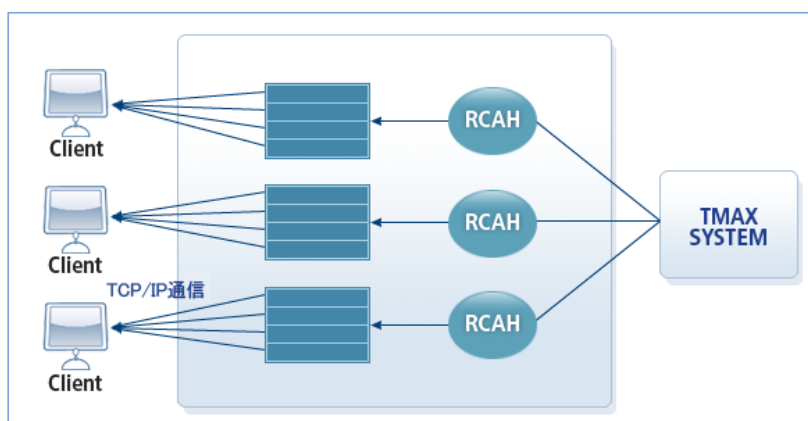
### 1.3.2. クライアント・モード

クライアント・モードは、RCAリモート・ソケット・プログラムから接続要求を待ち、RCAHの各スレッドで接続を要求する方式です。

リモート・ソケット・プログラムと接続が確立されたら、サーバー・モードもクライアント・モードも、RCA動作は同じです。ただし、クライアント・モードの場合は、RCALはリモート接続要求を待ちません。各スレッド・メイン・プログラムはサーバー・モードとクライアント・モードが異なります。

以下の図は、クライアント・モードでのRCAHのプロセス・フローです。

【図 1.2】 クライアント・モードでのプロセス・フロー



## 1.4. 特徴

RCAの特徴は、マルチ・ポートをサポートするということです。RCAは様々な形式のクライアントをサポートするために最大32個までポートを指定できます。したがって、開発者はポート別ロジックを構成することで、1つのプロセスより柔軟なプログラムを作成できます。

以下は、マルチ・ポートを指定する方法です。(デフォルト値: 8899)

```
RCA_PORT="9000, 9001, 9002, 9003"
```

プログラミングの際は、thrinit()、thrmain()、thrdone()の引数として渡される値がRCAINFO構造体(\$TMAXDIR/usrinc/rca.hの以下参照)のポインターであり、この構造体が指すポート番号を基準にしてポート別のロジックを構成することができます。

```
/* ----- type definition ----- */
typedef struct {
    int fd;
    int portno;
    int count;
    int status1;
    int status2;
    void *user_data;
    void *system_data;
} *RCAINFO;
```

以下は、構造体フィールドについての説明です。

構造体フィールド	説明
fd	クライアントが接続して生成されたTCP/IPソケットです
portno	使用されたポート番号です。この場合は、9000、9001、9002、9003のいずれかの値です
count	スレッドが呼び出された回数です

注

本書で説明していないフィールドは、RCA内部で使われるフィールドなので、ユーザーが操作してはなりません。

## 第2章 使用と管理

本章ではRCAシステムを使用するためのTmax環境設定の手順と管理ツールの使用方法について説明します。

### 2.1. 環境設定

提供されるファイルをそれぞれのディレクトリに適切な名前に変更してコピーします。下の例はTmax 3.7.6を基準にしています。

Directory	ファイル名
\$TMAXDIR/bin	rcal
\$TMAXDIR/bin	rcastat
\$TMAXDIR/bin	rcakill
\$TMAXDIR/lib	librcah.so
\$TMAXDIR/usrinc	rca.h

### 環境変数の設定

RCAで使用する環境変数は以下のとおりです。各プラットフォームに合わせて設定します。

環境変数	説明
RCA_DIR	RCAHがインストールされ実行されるホームディレクトリの絶対パスを指定します。RCALはRCA_DIRで定義されたディレクトリでRCAHを探して実行させます
RCA_SHMKEY	RCALとRCAHプロセス間で情報共有のために使用する共有メモリーKey値です(デフォルト値 : 74565)
RCA_NRCAH	RCAHモジュールの数を指定します(デフォルト値: 4、最大値: 500)
RCA_NTHR	RCAHごとに管理するスレッド数を指定します(デフォルト値: 32、最大値 :60)
RCA_PORT	RCALが従来の通信プログラム(クライアント)から接続を受けるポート番号です。ポートは最大32個まで指定できます(デフォルト値 : 8899)  (例: RCA_PORT="9000 9001 9002 9003")
LD_LIBRARY_PATH	ローカル・モードの場合にはLD_LIBRARY_PATHにTmax Library Pathを追加します。一方、リモート・モードの場合はlibrcah.soが存在するディレクトリをLD_LIBRARY_PATHに指定します

以下は、ローカル・モードでkorn shellを使用する場合、.profile内にRCA関連環境変数を設定する例です。

<.profile>

```
export RCA_DIR = /usr/tmax/appbin
export RCA_SHMKEY = 79800
export RCA_PORT = 2001
export RCA_NRCAH = 5
export RCA_NTNR = 25
export LD_LIBRARY_PATH = $LD_LIBRARY_PATH:$TMAXDIR/lib
```

## 環境設定ファイルの修正

RCAをローカル・モードで運用する場合には、以下のようにconfig.mファイルのSERVERセクションに追加します。

<config.m >

```
*SVRGROUP
svg1      NODENAME = "tmax1"

*SERVER
RCASVR    SVGNAME = svg1,
          TARGET = "rcal",
          SVRTYPE = SYS_SVR,
          CLOPT = "-n rcah -f /tmax/rca.env", MIN = 1
```

## rcadモジュールの作成および準備

カスタマイズされたRCAHをRCA\_DIR環境変数に定義されたディレクトリーにコピーします。

## 2.2. システムの使用

RCAを使用するためには、まずRCAHを作成する必要があります。ユーザーは、thrinit()、thrmain()、thrdone()のルーチンを実装してRCAHライブラリーとリンクし、RCAH実行ファイルを作成します。このように作成された実行ファイルはRCA\_DIR環境変数に設定したディレクトリーに存在する必要があります。

関数	説明
thrinit()	スレッドが初めて起動されるときに呼び出されます。一般的にthrinit()でTmaxシステムとの接続を処理し、thrdone()ではTmaxシステムとの接続の解除を処理します
thrmain()	新しいクライアント・ソケットが接続されるとき、該当するソケットFDとポート番号を引数として使用して呼び出されます。(\$TMAXDIR/usrinc/rca.hのRCAINFO構造体を参照)



関数	説明
thrdone()	スレッドが終了するときに呼び出されます

RCAを終了する方法は、使用するモードに応じて違いがあります。RCAがTmaxシステムと同じノードで運用されるローカル・モードでは、rcakillを使用してRCAを終了するか、または、Tmaxシステム・コマンドのtmddownを使って一緒に終了することもできます。一方、RCAがTmaxシステムと別のノードで運用されるリモート・ノードでは、RCAを別のコマンドを使って起動させる必要があるため、強制終了あるいはrcakillを使用して終了します。

#### 注

RCAHをコンパイルする際は、usrinc/rca.hが他のTmaxヘッダー・ファイルより先にインクルードされる必要があります。

## 2.2.1. ローカル・モード

ローカル・モードはTmaxシステムとRCAモジュールが同じノードにあり、統合管理が必要な場合に使用します。

RCALは1つのTmaxサーバー形式で登録され、Tmaxシステムと接続されます。RCALは別途の実行・終了コマンドを使用せずに、tmbootやtmddownで起動および終了され、Tmaxシステムとはパイプ通信を行います。

tmbootコマンドによって起動されたRCALは、環境ファイルに指定されたファイルからRCA\_DIR環境変数を読み込んで、通信用パイプとRCAH実行ファイルの場所を検索します。RCA\_DIR環境変数が定義されていない場合は、該当するノードのTmaxシステムのAPPDIRをRCA\_DIRとして使用します。したがって、RCALはRCA\_DIRディレクトリーに通信用パイプを生成し、RCAHを起動させます。RCAの登録はTmaxシステム・サーバー・プロセスと同じ方法で処理し、Tmaxシステム・クライアントと同じ方法で管理します。

ローカル・モードでRCAはTmaxシステムのサーバープロセス形式で制御できるため管理が便利です。また、Tmaxシステムとストリーム・パイプを通じて接続されるため、リモート・モードの場合より迅速なデータ送受信が可能です。RCALを登録する方法は一般的なサーバー・プロセスを登録する方法と同様です。

以下のように、rcalという名前を持つSYS\_SVRタイプのサーバー・プロセスを登録します。モードがローカルであることは自動的に認識されます。

```
*SERVER
rcal      SVGNAME = svg1,
          SVRTYPE = SYS_SVR,
          CLOPT = "-n rcah -f /tmax/rca.env", MIN = 1
```

## 注

RCALプロセスは2つ以上のプロセスが起動されると同じポートを使用するため、1つのプロセスのみ起動できます。

tmbootとtmdownコマンドを実行するとき、サーバー名をrcal以外の名前にする場合は、以下のように指定します。

```
*SERVER
RCASVR      SVGNAME = svg1, TARGET = "rcal",
             SVRTYPE = SYS_SVR,
             CLOPT = "-n rcah -f /tmax/rca.env", MIN = 1
```

以下は、CLOPT項目の設定オプションについての説明です。

オプション	説明
[-n]	RCAH実行ファイルの名前を指定します(デフォルト値 : rcah)
[-f]	RCAで使用する環境ファイルを指定します。デフォルト値はありません。  [-f]オプションで設定されるファイルはRCAを動作させるための基本的な情報であり、正常な動作を保証するためにはRCAを起動する前に設定する必要があります。 詳細については、「 <a href="#">2.1. 環境設定</a> 」を参照します

## 2.2.2. リモート・モード

リモート・モードはRCAモジュールをTmaxシステムとは別途に管理したい場合に使用する方式であり、一般Tmaxクライアントモジュールと同じ方式(TCP/IPソケット)でTmaxに接続します。TMAX\_HOST\_ADDR、TMAX\_HOST\_PORT、TMAX\_BACKUP\_ADDR、TMAX\_BACKUP\_PORT、TMAX\_CONNECT\_TIMEOUTなどを使用して接続します。リモート・モードではRCALを別途に起動させる必要があります。

以下はリモート・モードの実行方法です。ローカル・モードには適用できません。

### ● 使用方法

```
$ rcal [-n RCAH_NAME] [-f ENV_FILE] [-m MODE]
```

オプション	説明
[-n RCAH_NAME]	RCAH実行ファイルの名前を指定します(デフォルト値 : rcah)
[-f ENV_FILE]	RCAで使用する環境ファイルを指定します。デフォルト値はありません
[-m MODE]	ローカル・モードとリモート・モードのいずれかを選択します  - 0: リモート・モード(デフォルト値)

オプション	説明
	– 1: ローカル・モード(Tmax 3.7.6ではサポートしていません)

---

#### 参考

[[-n](#)]と[[-f](#)]オプションの使用方法はローカル・モードの場合と同じです。環境変数については、[「2.1. 環境設定」](#)を参照してください。

---

## 2.3. システム管理

システムを管理するための起動、終了、監視、障害対策の方法について説明します。

### 2.3.1. 起動および終了

RCAはマルチ・スレッド・クライアントとして起動され処理されます。リモート・モードの場合は管理者が直接起動しなければなりませんが、ローカル・モードの場合にはTmaxシステムのサーバー・プロセス形式で登録されるため、Tmaxシステムと一緒に起動および終了できます。一般的なサーバー・プロセスと同様、[[-s](#)]オプションを使用して起動および終了することもできます。

RCAの起動と終了は、**tmboot**と**tmdown**コマンドを使用します。以下は、コマンドの使用例です。

```
$ tmboot -s rcal
$ tmdown -s rcal
```

RCAモジュールは起動時にthrinit()ルーチンを呼び出します。一般的にthrinit()で初期化内容を設定します。thrinit()で設定する初期化の主な作業は、Tmaxシステムと接続を確立することです。したがって、RCAHに属するそれぞれのスレッドは起動と同時にTmaxシステムと接続を確立し、クライアントの要求が発生する時点ではサービスの処理だけ行います。実際のサービス処理はthrmain()で設定します。

RCAHを作成してから5秒が過ぎてもRCALにいかなるRCAHも登録しない場合など、RCALがRCAHの起動に失敗すると、RCALがエラーメッセージとともに終了されます。(エラー番号[RCAL0002])

RCAモジュールが終了すると、thrdone()ルーチンを呼び出します。このルーチンではリソースの解除といった最終的な作業を処理します。thrdone()で設定するリソース解除の代表的な場合は、Tmaxシステムとの接続を解除することです。注意すべきことは、Tmaxシステムを終了させる場合は、thrdone()でTmaxシステムに関連する関数の呼び出しを実行できないということです。TmaxシステムでRCAはクライアント・プロセスとして管理されるため、Tmaxシステムは終了時にRCAとの接続を別途管理しないからです。

RCAは障害対策をサポートするため、tmdownで正常にシステムをダウンさせた場合にも共有メモリーが削除されません。共有メモリーを削除する場合は、rcakillコマンドを使ってRCAを終了します。RCAの障害対策については、[「2.4. 障害対策」](#)の内容を参照してください。

## 2.3.2. 監視

### 2.3.2.1. rcastat

RCAはTmaxシステムではクライアント・プロセスとして管理されるため、Tmaxシステム管理ツールのtmadminでモニタリングできます。TmaxではRCAをモニタリングする`rcastat`という別途のツールを提供しています。このツールを使って管理者は現在のRCAの設定内容を確認したり、現在RCAに接続しているクライアントの数などをモニタリングすることができます。

- 使用方法

```
rcastat [-h] [-p pid] [-n rcah_no] [-k shmkey]
```

項目	説明
[-h]	コマンドのヘルプ・オプションです
[-p pid]	pidにより特定のRCAHの状態情報を出力します。  pidを指定していない場合、RCAで動作しているすべてのRCAHの情報を出力します
[-n rcah_no]	RCAH番号により特定のRCAHの状態情報を出力します。  rcah_noを指定していない場合、RCAで動作しているすべてのRCAHの情報を出力します
[-k shmkey]	RCAが使用する共有メモリー・キー値を設定します。  共有メモリー・キー値を指定しない場合は、ユーザー・システム環境変数に設定したRCA_SHMKEY値を読み込みます。システム環境変数にも設定していない場合は、デフォルト値の74565が使用されます。誤ったキー値を入力した場合には、正しい情報を出力できません

- 例

```
-----  
rca_dir: /user/tmax/server/  
rca_mode: Local  
rca_port: 8123  
rcal_name: rca  
rcal_pid: 1722  
rcah_name: rcah  
shmkey: 74565, shmsize: 4832  
#rcah: 2, #thread per rcah: 60  
-----  
rcah_no    pid    #client
```

```

-----
0      1724      0
1      1723      0
-----

```

出力項目名	説明
rca_dir	RCAH実行ファイルと通信用パイプが生成されるディレクトリーです
rca_mode	現在使用されているモード情報です
rca_port	使用可能なポート番号です
rcal_name	RCALの名前です
rcal_pid	RCALのpidです
rcah_name	RCAHの名前です
shmkey	RCAで使用している共有メモリー・キー値です
shmsize	共有メモリーのサイズです
#rcah	使用されているRCAの数です
#thread per rcah	1つのRCAH当たり起動されているスレッド数です
rcah_no	RCA内部で管理される番号であり、それぞれのRCAHごとに番号が振られます
pid	RCAに起動されているRCAH別のpidです
#client	クライアントの数です

以下は、コマンドを使って照会したシステムの設定情報です。

```

$ rcastat
    rcastat: RCA_SHMKEY env is not set, using default shmkey (74565)

```

## 2.4. 障害対策

- RCAH障害

開発者が直接作成したプログラムによって生成されるRCAHはプログラムバグによってダウンされる場合があります。

RCAHにクライアントが接続している状態でRCAHがダウンすると、RCALがRCAHを再起動します。ただし、RCAHにクライアントが接続していない状態でRCAHがダウンすると再起動されません。これは最も頻繁に発生する問題であり、このようなRCAHのダウンは、開発者が作成したプログラムが原因なので、RCAHに対するデバッグが必要です。再起動されたRCAHが、失敗した業務を再実行することはありません。

- RCAL障害

管理者のミスでRCALを終了する場合(tmdown)、RCAHはクライアントが接続されている限り実行を続けます。

一方、クライアントが接続していないRCAHは、RCALが終了されると自動的に終了されます。RCAが使用する共有メモリーはRCALが終了しても削除されないで、これを削除する場合は、ユーザーがipcrmを使用して直接削除します。ただし、rcakillを使用してRCAを終了する場合は、共有メモリーが自動的に削除されます。RCAH内のスレッドがダウンされた場合は再起動します。

クライアントとの接続が解除され業務の実行が完了すると、RCAHも終了されます。ただし、RCALに障害が発生した場合、新しいクライアントは接続できません。この場合には、RCALを再起動してください。

## 2.4.1. rcakill

RCAを終了したい場合、RCAが使用しているリソースを削除するために使用します。

- 使用方法

```
rcakill [-h] [-p pid] [-n rcah_no] [-k shmkey]
```

項目	説明
[-h]	コマンドのヘルプ・オプションです
[-p pid]	pidにより特定のRCAHの状態情報を削除します
[-n rcah_no]	RCAH番号により特定のRCAHの状態情報を削除します
[-k shmkey]	RCAが使用する共有メモリーのキー値の情報を削除します

## 第3章 APIとエラーメッセージ

本章では実際にアプリケーションを開発するためのAPIの使用方法和エラーメッセージを紹介します。

### 3.1. API

以下はRCA APIの一覧です。

API	説明
<a href="#">tpsetfd</a>	ソケットFDをUCSプロセスのスケジューラに登録する関数です
<a href="#">tpclrfd</a>	UCSプロセス内部のfdsetのソケットFDをオフにするために使用する関数です
<a href="#">tpissetfd</a>	UCSサーバー・プロセスでソケットFDにデータが受信したかをチェックする関数です
<a href="#">tpschedule</a>	UCSサーバー・プロセスでデータの受信を待つ関数です
<a href="#">tpuschedule</a>	UCSサーバー・プロセスでデータの受信を入力した時間の間待つ関数です
<a href="#">tpremoteconnect</a>	リモートとTCP/IPで接続する関数であり、特定の時間の間接続を試みます
<a href="#">tpgetrcahseqno</a>	tpgetsvrseqno APIと同様にRCAHプロセス番号を返す関数です
<a href="#">tpgetrcainfo</a>	RCAHのスレッド情報を照会する関数です

#### 3.1.1. tpsetfd

ソケットFDをUCSプロセスの外部ソケット・スケジューラに登録する関数です。UCS方式プロセスを使用したソケットFDを起動する際に使用されます。UCSスケジューラは、TMM、CLHだけでなく、該当ソケットFDに到着したメッセージも一緒にチェックします。ユーザーが指定したソケットにメッセージが到着した場合、tpschedule()は特に処理をすることなく、正常結果(UCS\_USER\_MSG)を返します。また、どのソケットにメッセージが到着したかを確認するためには、tpissetfd()を使用します。

- プロトタイプ

```
#include <ucs.h>
int tpsetfd (int fd)
```

- パラメータ

パラメータ	説明
fd	登録するソケットFDを設定します

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tpsetfd()が正常に実行されなかった場合、tperrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TPESYSTEM]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TPEOS]	運用システムにエラーが発生した場合です

- 例

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <usrinc/ucs.h>
...
#define SERV_ADDR "168.126.185.129"
#define SERV_PORT 1500

int fd_read(int, char *, int);
extern int errno;

int usermain(int argc, char *argv[])
{
    ...
    int listen_fd, n, newfd;
    struct sockaddr_in my_addr, child_addr;
    socklen_t child_len;
    buf = tpalloc("STRING", NULL, 0);
    if (buf == NULL){
        error processing
    }

    memset((void *)&my_addr, NULL, sizeof(my_addr));
    memset((void *)&child_addr, NULL, sizeof(child_addr));
    listen_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_fd == -1){
        error processing
    }
}
```



```

}

my_addr.sin_family = AF_INET;
inaddr = inet_addr(SERV_ADDR);
my_addr.sin_port = htons((unsigned short)SERV_PORT);

if (inaddr != -1){
    memcpy((char *)&my_addr.sin_addr, (char *)&inaddr, sizeof(inaddr));
}
ret = bind(listen_fd, (struct sockaddr *)&my_addr, sizeof(my_addr));
if (ret == -1){
    error processing
}
ret = listen(listen_fd, 5);
if (ret == -1){
    error processing
}

ret = tpsetfd(listen_fd);
if (ret == -1){
    error processing
}
...

while(1) {
    n = tpschedule(10);
    ...
    if (n == UCS_USER_MSG){
        if (tpissetfd(listen_fd)) {
            child_len = sizeof(child_addr);
            newfd = accept(listen_fd, &child_addr, &child_len);
            if (newfd == -1){
                error processing
            }

            ret = tpsetfd(newfd);
            if (ret == -1){
                error processing
            }
        }

        if (tpissetfd(newfd)){
            /* ソケットからバッファを読み込みます */
            fd_read(newfd, buf, 1024);
            ret = tpcall("SERVICE", (char *)buf, sizeof(buf), (char **)&buf,
                        (long *)&rlen, TPNOFLAGS);
            if (ret == -1){

```

```

        error processing
    }
    ...
    tpclrfd(newfd);
    close(newfd);
}
...
}
return 1;
}

```

- 関連関数

tpclrfd(), tpissetfd()

### 3.1.2. tpclrfd

UCS方式プロセス内部のfdsetのソケットFDをオフにするのに使用される関数です。UCS方式サーバー・プロセスの外部ソケットをスケジューリングする場合に使用します。

- プロトタイプ

```

#include <ucs.h>
int tpclrfd (int fd)

```

- パラメータ

パラメータ	説明
fd	オフにする内部fdsetのソケットです

- 戻り値

戻り値	説明
1	関数の呼び出しに成功した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tpclrfd()が正常に実行されなかった場合、tperrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TPESYSTEM]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます

エラーコード	説明
[TPEOS]	運用システムにエラーが発生した場合です

● 例

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <usrinc/ucs.h>
...
#define SERV_ADDR "168.126.185.129"
#define SERV_PORT 1500

int fd_read(int, char *, int);
extern int errno;

int usermain(int argc, char *argv[])
{
    ...
    int listen_fd, n, newfd;
    struct sockaddr_in my_addr, child_addr;
    socklen_t child_len;
    buf = tpalloc("STRING", NULL, 0);
    if (buf == NULL){ error processing }

    memset((void *)&my_addr, NULL, sizeof(my_addr));
    memset((void *)&child_addr, NULL, sizeof(child_addr));
    listen_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_fd == -1){ error processing }
    my_addr.sin_family = AF_INET;
    inaddr = inet_addr(SERV_ADDR);
    my_addr.sin_port = htons((unsigned short)SERV_PORT);
    if (inaddr != -1){
        memcpy((char *)&my_addr.sin_addr, (char *)&inaddr, sizeof(inaddr));
    }

    ret = bind(listen_fd, (struct sockaddr *)&my_addr, sizeof(my_addr));
    if (ret == -1){ error processing }
    ret = listen(listen_fd, 5);
    if (ret == -1){ error processing }

    tpsetfd(listen_fd);
    ...
    while(1) {
        n = tpschedule(10);
```

```

...
if (n == UCS_USER_MSG){
if (tpissetfd(listen_fd)) {
    child_len = sizeof(child_addr);
    newfd = accept(listen_fd, &child_addr, &child_len);
    if (newfd == -1){ error processing }
    tpsetfd(newfd);
}

if (tpissetfd(newfd)){
    /* ソケットからバッファを読み込みます */
    fd_read(newfd, buf, 1024);
    ret = tpcall("SERVICE", (char *)buf, sizeof(buf), (char **)&buf,
                (long *)&rlen, TPNOFLAGS);
    if (ret == -1){ error processing }
    ...
    ret = tpclrfd(newfd);
    if (ret == -1){ error processing }
    close(newfd);
}
...
}
return 1;
}

```

- 関連関数

tpissetfd()

### 3.1.3. tpissetfd

UCSプロセスでソケットFDにデータが到着したかどうかをチェックする関数です。UCS方式サーバー・プロセスの外部ソケットのスケジューリングに使用されます。

- プロトタイプ

```

#include <ucs.h>
int tpissetfd (int fd)

```

- パラメータ

パラメータ	説明
fd	テストするfdset内部のFDを設定します

- 戻り値

戻り値	説明
正数	メッセージが到着した場合です
0	メッセージが到着しなかった場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tpissetfd()が正常に実行されなかった場合、tperrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TPESYSTEM]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TPEOS]	運用システムにエラーが発生した場合です

- 例

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <usrinc/ucs.h>
...

#define SERV_ADDR "168.126.185.129"
#define SERV_PORT 1500

int fd_read(int, char *, int);
extern int errno;

int usermain(int argc, char *argv[])
{
    ...
    int listen_fd, n, newfd;
    struct sockaddr_in my_addr, child_addr;
    socklen_t child_len;

    buf = tpalloc("STRING", NULL, 0);
    if (buf == NULL){ error processing }

    memset((void *)&my_addr, NULL, sizeof(my_addr));
    memset((void *)&child_addr, NULL, sizeof(child_addr));

    listen_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_fd == -1){ error processing }
```

```

my_addr.sin_family = AF_INET;
inaddr = inet_addr(SERV_ADDR);
my_addr.sin_port = htons((unsigned short)SERV_PORT);
if (inaddr != -1)
    memcpy((char *)&my_addr.sin_addr, (char *)&inaddr, sizeof(inaddr));

ret = bind(listen_fd, (struct sockaddr *)&my_addr, sizeof(my_addr));
if (ret == -1){ error processing }
ret = listen(listen_fd, 5);
if (ret == -1){ error processing }

tpsetfd(listen_fd);
...

while(1) {
    n = tpschedule(10);
    ...
    if (n == UCS_USER_MSG){
        if (tpissetfd(listen_fd)) {
            child_len = sizeof(child_addr);
            newfd = accept(listen_fd, &child_addr, &child_len);
            if (newfd == -1){ error processing }
            tpsetfd(newfd);
        }

        if (tpissetfd(newfd)){
            /*ソケットからバッファを読み込みます*/
            fd_read(newfd, buf, 1024);
            ret = tpcall("SERVICE", (char *)buf, sizeof(buf), (char **)&buf,

                        (long *)&rlen, TPNOFLAGS);
            if (ret == -1){ error processing }
            ...
            tpclrfd(newfd);
            close(newfd);
        }
        ...
    }
}
return 1;
}

```

- 関連関数

tpissetfd(), tpsetfd()

### 3.1.4. tpschedule

UCS形式のサーバー・プロセスでのみ使用される関数で、UCSサーバー・プロセスでデータの到着を待機します。最大タイムアウト時間の間待機し、その間にデータが到着した場合は即時に返します。

tpschedule()関数は、データの到着時に該当するサービスが自動的に実行された後に返されます。そのため、データが到着後にユーザーが任意でサービスを実行してはいけません。

---

#### 注

サービスは常にシステムによって実行されるため、UCS形式のサービス・プログラムでもこの点は注意してください。

---

#### ● プロトタイプ

```
#include <ucs.h>
int tpschedule(int timeout)
```

#### ● パラメータ

パラメータ	説明
timeout	待機する時間を秒単位で入力します – -1: データが到着したかどうかをチェックのみ行った後、すぐに返します – 0: データが到着するまで無限に待機します

#### ● 戻り値

戻り値	説明
正の整数	関数の実行に成功してデータが到着した場合です
-1	タイムアウト時間までにデータが到着していない場合です。あるいは、関数が実行に失敗してエラーが発生した場合です。タイムアウト時間までデータが到着しなかった場合は-1を返し、tperrnoに13番(TPETIME)が設定されます。それ以外の場合、tperrnoにエラーコードが設定されます

#### ● エラー

tpschedule()が正常に実行されなかった場合、tperrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TPESYSTEM]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます

エラーコード	説明
[TPEOS]	運用システムにエラーが発生した場合です
[TPETIME]	タイムアウト時間までにデータが到着していない場合です

- 例

```
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/ucs.h>
int usermain(int argc, char *argv[])
{
    ...
    while(1)
    {
        ...
        tpschedule(3);
        ret = tpcall("SERVICE", (char *)buf, strlen(buf), (char **)&buf,
                    (long *)&rlen, TPNOFLAGS);
        if (ret == -1) { error processing}
        ...
    }
}
```

- 関連関数

tpsleep(), tp\_sleep(), tp\_usleep()

## 3.1.5. tpuschedule

UCSサーバー・プロセスで、データの到着をマイクロ秒単位で入力した時間の間待機する関数です。tpuschedule()はUCS形式のサーバー・プロセスでのみ使用可能な関数で、最大タイムアウト時間の間待機し、決められた時間内にデータが到着すれば即時返します。

- プロトタイプ

```
#include <ucs.h>
int tpuschedule (int timeout)
```

- パラメータ

パラメータ	説明
timeout	待機する時間をマイクロ秒単位で入力します - -1: データが到着したかどうかチェックのみ行い、すぐに終了します



パラメータ	説明
	– 0: データが到着するまで待機します

- 戻り値

戻り値	説明
0	タイムアウト時間までにデータが到着していない場合です
正の整数	タイムアウト時間までにデータが到着した場合です
-1	関数の呼び出しに失敗した場合です。tperrnoにエラーコードが設定されます

- エラー

tpuschedule()が実行されなかった場合、tperrnoに以下の値のいずれかが設定されます。

エラーコード	説明
[TPESYSTEM]	Tmaxシステムにエラーが発生した場合です。詳細情報はログファイルに記録されます
[TPEOS]	運用システムにエラーが発生した場合です

- 例

```
...
#include <stdio.h>
#include <usrinc/atmi.h>
#include <usrinc/ucs.h>

int usermain(int argc, char *argv[])
{
    ...
    while(1)
    {
        ...
        tpuschedule(3000000);
        ret = tpcall("SERVICE", (char *)buf, strlen(buf), (char **)&buf,
                    (long *)&rlen, TPNOFLAGS);
        if (ret == -1) { error processing }
        ...
    }
}
```

- 関連関数

tpschedule()

### 3.1.6. tpremoteconnect

リモート・ホストとTCPで接続する関数です。secが与えられると、その時間の間、接続を試みます。その時間を超えれば、接続エラーが発生します。

- プロトタイプ

```
#include <ucs.h>
int tpremoteconnect(char *host, int portno, int sec)
```

- 戻り値

戻り値	説明
0	ソケット番号です
-1	関数の呼び出しに失敗した場合です。tpernoにエラーコードが設定されます

### 3.1.7. tpgetrcahseqno

Tmax APIのtpgetsvrseqno APIと同様に、RCAHプロセス番号を返す関数です。この番号はRCALが順次に与えた番号です。

- プロトタイプ

```
#include <ucs.h>
int tpgetrcahseqno()
```

- 戻り値

戻り値	説明
0	RCAHプロセスの順序番号です

### 3.1.8. tpgetrcainfo

RCAHのスレッド情報を表示する関数です。

- プロトタイプ

```
#include <ucs.h>
void * tpgetrcainfo()
```

- 戻り値

RCAヘッダー・ファイルのRCAINFOに対するポインターを返します。

## 3.2. エラーメッセージ

RCAは基本的にTmaxクライアントとして動作するため、Tmaxシステムで提供する関数を使用する場合に発生するエラーメッセージは、『Tmax アプリケーション開発ガイド』を、エラーメッセージについては、『Tmax メッセージリファレンスガイド』を参照してください。



## 第4章 RCAHの例

本章では、RCAHカスタマイズと例について説明します。

### 4.1. RCAHカスタマイズ

各プロジェクトごとに通信プログラム(Client)とRCA間の通信プロトコルが異なるため、プロトコルを合わせる必要があります。RCAHモジュールでこれ进行处理する`int thrmain(RCAINFO info)`を開発者がカスタマイズします。この関数はスレッドの一部であるため、MT-Safeを満たすように作成する必要があります。

RCAHはRCA\_NTHRに定義された値のスレッドを作成し、クライアントから要求があるときスレッドプールから1つのスレッドを割り当てて基本的な環境を設定し、クライアントの要求が発生した時点でユーザーが作成した`thrmain()`を呼び出す方法で処理します。`thrmain()`の実行が終了すると、クライアントとの接続が切れ、スレッドは再びスレッドプールに返されます。

---

#### 注

グローバル変数を使用して書き込みする場合は、RCAHがマルチスレッドであるためロック処理をする必要があります。`tmaxreadenv()`を使用する場合、メモリー損失が発生するので使用できません。

---

### 4.2. 例

以下は、Solaris 2.7 32ビットのマシンでテストした例です。TMAXDIR環境変数がシステムに設定されていると想定します。

#### 4.2.1. Makefile

以下は、Makefileの例です。

```
TARGET = rcah
LIBS    = -lrcah -lpthread -lsocket
CFLAGS  = -I$(TMAXDIR)

APOBJS  = $(TARGET).o

TMAX_INCDIR = $(TMAXDIR)/usrinc
TMAX_BINDIR = $(TMAXDIR)/bin
TMAX_LIBDIR = $(TMAXDIR)/lib
OBSJS      = $(APOBJS)
```

```

.c.o:
$(CC) $(CFLAGS) -c $<

$(TARGET): $(APOBJS)
    $(CC) $(CFLAGS) -L$(TMAX_LIBDIR) -o $(TARGET) $(OBJS) $(LIBS) $(USERLIBS)

clean:
-rm -f *.o core $(TARGET)

```

## 4.2.2. プログラム

以下は、プログラムのヘッダー・ファイルとソース・ファイルの例です。

### <rca.h>

```

/* ----- usrinc/rca.h ----- */
/*
    */
/*      Copyright (c) 2002 TmaxSoft Co., Ltd      */
/*      All Rights Reserved      */
/*
    */
/* ----- */

#ifndef _TMAX_RCA_H
#define _TMAX_RCA_H
#ifndef _TMAX_MTLIB
#define _TMAX_MTLIB 1
#endif

#include <usrinc/tmaxapi.h>
#ifndef _WIN32
#define __cdecl
#endif

/* ----- type definition ----- */
typedef struct {
    int fd;
int portno;
    int count;
    int status1;
    int status2;
    void *user_data;
    void *system_data;
} *RCAINFO;

```

```

#if defined (__cplusplus)
extern "C" {
#endif

#ifdef _TMAX_KERNEL
int thrmain(RCAINFO);
int thrinit(RCAINFO);
int thrdone(RCAINFO);
#endif

#if defined (__cplusplus)
}
#endif

#endif

```

## <rcah.c>

```

#include      <stdlib.h>
#include      <string.h>
#include      <unistd.h>
#include      <netdb.h>
#include      <sys/types.h>
#include      <sys/socket.h>
#include      <sys/un.h>
#include      <netinet/in.h>
#include      <arpa/inet.h>
#include      <pthread.h>
#include      <errno.h>
#include      <usrinc/rca.h>

struct msg {
    int      len;
    char      svcname[16];
};

int start_flag;

int thrinit(RCAINFO info)
{
    int n;

    n = tpstart(NULL);
    if (n < 0) {
        printf("RCA_%d: tpstart fail, tperrno = %d, errno = %d\n",

```

```

        pthread_self(), tperrno, errno);
    return -1;
}

start_flag = 1;
printf("RCA_%d: thrinit ok\n", pthread_self());
return 1;
}

int thrdone(RCAINFO info)
{
    if (start_flag) {
        start_flag = 0;
        tpend();
    }
    printf("RCA_%d: thrdone ok\n", pthread_self());
    return 0;
}

int thrmain(RCAINFO info)
{
    struct msg header;
    int n;
    long len;
    char *sndbuf;
    char buf[8192];

    buf[0] = 0;

    n = read(info->fd, &header, sizeof(struct msg));
    if (n != sizeof(struct msg)) {
        printf("RCA_%d: read header fail = %d\n", pthread_self(), n);
        return -1;
    }

    len = ntohl(header.len);

    n = read(info->fd, buf, len);
    if (n != len) {
        printf("RCA_%d: read data fail = %d\n", pthread_self(), n);
        return -1;
    }
}

```



```

/* ----- */

if (!start_flag) {
    n = tpstart(NULL);
    if (n < 0) {
        printf("RCA_%d: tpstart fail, tperrno = %d\n", pthread_self(),
            tperrno);
        return -1;
    }
    start_flag = 1;
}

sndbuf = (char *)tpalloc("CARRAY", NULL, len + 1);
if (sndbuf == NULL) {
    printf("RCA_%d: tpalloc fail, tperrno = %d\n", pthread_self(), tperrno);

    return -1;
}

memcpy(sndbuf, buf, len);
n = tpcall(header.svcname, sndbuf, len, &sndbuf, &len, 0);
if (n < 0) {
    printf("RCA_%d: tpcall fail, tperrno = %d\n", pthread_self(), tperrno);

    tpfree(sndbuf);
    return -1;
}
memcpy(buf, sndbuf, len);
header.len = htonl(len);
tpfree(sndbuf);

/* ----- */
write(info->fd, &header, sizeof(struct msg));
write(info->fd, buf, len);

return 0;
}

```



# 索引

## シンボル

#client, 11  
#rcah, 11  
#thread per rcah, 11

## C

CLOPTオプション  
[-f], 8  
[-n], 8

## P

pid, 11

## R

RCA, 1  
rca\_dir, 11  
rca\_mode, 11  
rca\_port, 11  
RCAH, 2  
rcah\_name, 11  
rcah\_no, 11  
RCAH障害対策, 11  
rcakill, 2, 12  
RCAL, 1  
rcal\_name, 11  
rcal\_pid, 11  
rcalオプション  
[-f], 8  
[-m], 8  
[-n], 8  
RCAL障害対策, 12  
rcastat, 2, 10  
rcastatオプション  
[-h], 10, 12  
[-k], 10, 12  
[-n], 10, 12

[-p], 10, 12  
RCAマルチ・ポート, 3  
RCA環境変数, 5

## S

shmkey, 11  
shmsize, 11

## T

tmboot, 9  
tmdown, 9  
tpclrfd, 16  
tpgetrcahseqno, 24  
tpgetrcainfo, 24  
tpissetfd, 18  
tpremoteconnect, 24  
tpschedule, 21  
tpsetfd, 13  
tpuschedule, 22

## か

環境設定ファイル, 6

環境変数

LD\_LIBRARY\_PATH, 5  
RCA\_DIR, 5  
RCA\_NRCAH, 5  
RCA\_NTHR, 5  
RCA\_PORT, 5  
RCA\_SHMKEY, 5

構造体フィールド

count, 4  
fd, 4  
portno, 4

クライアント・モード, 3

## さ

サーバー・モード, 2

システム使用ルーチン

thrdone(), 7  
thrinit(), 6  
thrmain(), 6

## ら

リモート・モード, 8

リモート・モード(Remote Mode), 1

ローカル・モード, 7

ローカル・モード(Local Mode), 1