T-Kernel Standard Extension仕様書

T-Kernel Standard Extension 1.00.02

2008年12月

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

T-Kernel Standard Extension 仕様書(Ver.1.00.02)

本仕様書の著作権はT-Engine フォーラムに属しています。
本仕様書の内容の転記、一部複製等はT-Engine フォーラムの許諾が必要です。
本仕様書に記載されている内容は今後改良等の理由でお断りなしに変更することがあります。

本仕様書に関しては下記にお問い合わせください。
T-Engine フォーラム事務局
〒141-0031 東京都品川区西五反田2-20-1 第28 興和ビル
YRP ユビキタス・ネットワーキング研究所内
TEL:03-5437-0572 FAX:03-5437-2399
E-mail:office@t-engine.org

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

目次

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

図表索引

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

API 索引

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

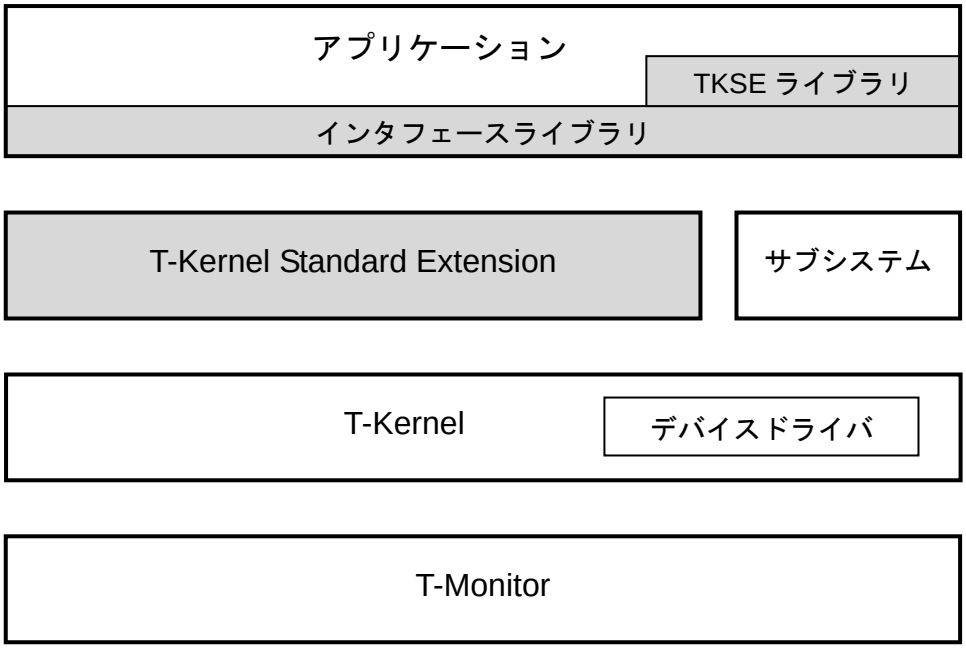
第1章　T-Kernel Standard Extension の概要

1.1 T-Kernel Standard Extension の位置付け

T-Kernel Standard Extension(以下 Standard Extension)は、T-Kernel の機能拡張プログラムである。

T-Kernel の機能を拡張し、より高度な OS の機能を実現するためのプログラムを T-Kernel Extension と呼ぶ。Standard Extension は標準仕様の T-Kernel Extension であり、T-Kernel にファイル管理やプロセス管理などの、大規模なシステムで一般的に必要とされる機能を追加する。

Standard Extension の T-Engine システム全体における位置付けを以下に示す。



[図 1] Standard Extension の位置付け

T-Kernel は組込みシステムを主な対象としたリアルタイム OS であり、T-Engine システムの中核となるものである。

Standard Extension は T-Kernel 上で動作する T-Kernel Extension であり、その主要な機能は T-Kernel サブシステムとして実装される。これらのサブシステムの拡張 SVC を C 言語の関数の形式で呼び出すためのライブラリをインタフェースライブラリと呼ぶ。また、Standard Extension の一部の機能は、サブシステムではなくアプリケーションに直接リンクする TKSE(T-Kernel Standard Extension)ライブラリとして提供される。

Standard Extension 上で動作する、ユーザが作成したプログラムをアプリケーションと呼ぶ。アプリケーションは Standard Extension の API(Application Programming Interface)を用いてそれぞれの機能を利用する。Standard Extension の API のうち、インタフェースライブラリを用いて呼び出す関数をシステムコール、TKSE ライブラリを用いて呼び出す関数をライブラリコールと呼ぶ。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

1.2 提供する機能

Standard Extension は以下の機能を提供する。

・メモリ管理
・プロセス／タスク管理
・プロセス間メッセージ
・グローバル名
・タスク間同期・通信
・標準入出力
・標準ファイル管理
・イベント管理
・デバイス管理
・時間管理
・システム管理
・共有ライブラリ

それぞれの機能の詳細、および API 仕様は別章で説明する。

1.3 対象とする動作環境

Standard Extension が動作する環境の条件として、T-Kernel のすべての機能を使用可能である必要がある。また、T-Kernel 単体では CPU の MMU(メモリ管理ユニット)が存在しない環境でも動作可能であるが、Standard Extension は MMU を必須とする。

Standard Extension が動作するためには、以下に示す T-Engine 標準デバイスドライバ仕様に準拠した T-Kernel デバイスドライバが必須である。

・システムディスクドライバ　… メモリ管理、プロセス／タスク管理、標準ファイル管理、標準入出力で使用

また、Standard Extension は以下のデバイスドライバも使用する。ただし、該当する機能を使用しない場合は、これらのデバイスドライバは不要である。

・時計（クロック）ドライバ　… 時間管理における RTC 時刻の取得・設定
・KB/PD ドライバ　… イベント管理における KB/PD イベント受信
・コンソール　… 標準入出力におけるコンソール入出力

上記の各デバイスドライバが他のドライバやサブシステム（PCMCIA カードマネージャなど）に依存している場

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

合は、それらも必要となる。

Standard Extension では、メモリ空間は MMU を用いて論理空間として使用される。このためデバイスドライバは、プロセスが確保した論理空間のバッファ領域にも正常にアクセスできる必要がある。ドライバは必要に応じて、タスク固有空間の切り替えや領域の常駐化、論理アドレスから物理アドレスへの変換などを行わなければならない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

第2章　T-Kernel Standard Extension の概念

2.1　プロセス

2.1.1　プロセスとは

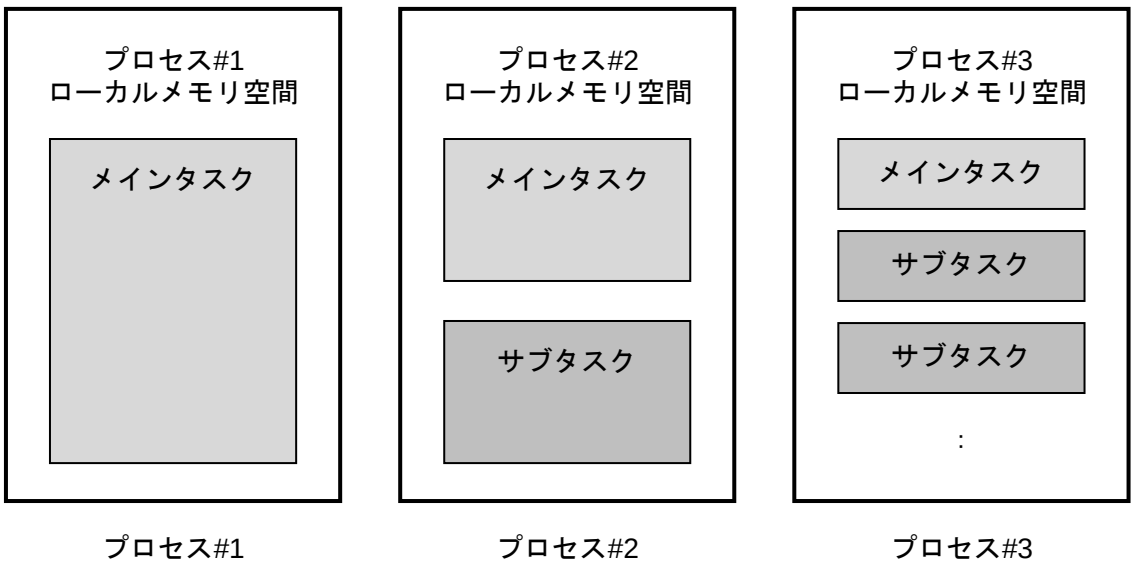
プロセスとは Standard Extension がプログラムを管理する単位である。１つのシステム上に、複数のプロセスを同時に存在させることができる。プロセスはそれぞれ独立したローカルメモリ空間や実行環境を持ち、他のプロセスと並行して動作する。

プロセスは、ファイルシステム上の実行プログラムファイルを、プロセス生成システムコールで読み込むことで生成される。生成したプロセスにはそれぞれ一意のプロセス ID が割り付けられ、この ID で個々のプロセスを識別する。プロセス ID は正の整数値をとる。

１つのプロセスは、１つ以上のタスクを持つ。タスクとはプログラムの実行単位であり、各タスクはタスク優先度に応じたスケジューリングにより動作する。

プロセス生成直後に実行可能状態となるタスクをメインタスクと呼ぶ。メインタスクはプロセスに必ず１つだけ存在し、メインタスクが終了すると直ちにプロセス全体が終了する。またメインタスクとは別に、タスク生成システムコールを呼び出すことでサブタスクを生成することができる。サブタスクは１つのプロセス内に複数生成することが可能である。サブタスクを終了してもプロセスは終了しない。これらのメインタスクとサブタスクを総称してタスクと呼ぶ。タスクは生成時に一意のタスク ID を割り付けられ、この ID で個々のタスクを識別する。タスク ID は正の整数値をとる。

同一プロセス内のタスクはローカルメモリ空間を共有する。



[図 2] プロセスとメインタスク・サブタスクの関係

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセスを生成した元のプロセスを親プロセス、生成されたプロセスを子プロセスと呼ぶ。すべてのプロセスは親プロセスを持つ。ただし、システム起動時に最初に生成される初期プロセスは、例外的に親プロセスを持たない。したがってシステム全体としては、初期プロセスをルートとする木構造のプロセス構造となる。

木構造の中のあるプロセスＡが終了した場合、プロセスＡの子プロセスの親プロセスは、プロセスＡの親プロセスに入れ替わり、全体として木構造は保たれる。例外として、初期プロセスが終了した場合、その子プロセスの親プロセスは存在しない状態となる。

2.1.2 プロセスのアドレス空間

プログラムからのアドレス指定によってアクセス可能な空間をアドレス空間と呼ぶ。32bit のアドレス空間は 0x00000000～0xFFFFFFFF までのアドレスを持ち、このアドレスにより各アドレスに対応付けられたメモリや I/O デバイスにアクセスすることができる。

アドレス空間には、システムのハードウェア設計時に定められる物理アドレス空間と、MMU などの機能を利用して仮想的に管理される論理アドレス空間が存在する。Standard Extension のプロセスは、通常は論理アドレス空間のみを使用する。

アドレス空間のうち、メモリに対応付けられた空間をメモリ空間と呼ぶ。ただし、Standard Extension は仮想記憶をサポートしているため、メモリ空間のすべてのアドレスに実際の物理メモリが割り当てられているとは限らない。

メモリ空間に実際のメモリを割り当ててアクセス可能とするには、Standard Extension のメモリ管理機能のシステムコール(tkse_get_mbk)またはライブラリ(malloc 系 API)を使用する。こうして割り当てられた、連続する論理アドレスを持つメモリのかたまりをメモリ領域と呼ぶ。プロセスは必要に応じてメモリ領域の割り当て・解放を行い、メモリに対するアクセスを実現する。

Standard Extension はファイルシステム上のページファイルを利用して仮想記憶管理を行う。これにより、実際の物理メモリサイズ以上のメモリ領域を使用することができる。メモリ領域のページイン・ページアウトは Standard Extension のメモリ管理が自動的に行うため、アプリケーションはメモリ領域が物理メモリ上に存在するかどうかを意識せずに使用することができる。また、リアルタイム性が必要とされるメモリアクセスについては、メモリを常駐指定することで対象領域のページアウトを禁止することができる。常駐指定したメモリ領域は常に物理メモリ上に存在する。

Standard Extension のメモリ空間は以下の３種類が存在する。

- ローカルメモリ空間
- 共有メモリ空間
- システムメモリ空間

ローカルメモリ空間は、プロセスごとに独立したアドレスと内容をもつメモリ空間である。プロセスの使用するコード領域およびデータ領域は、通常はローカルメモリ空間に配置される。

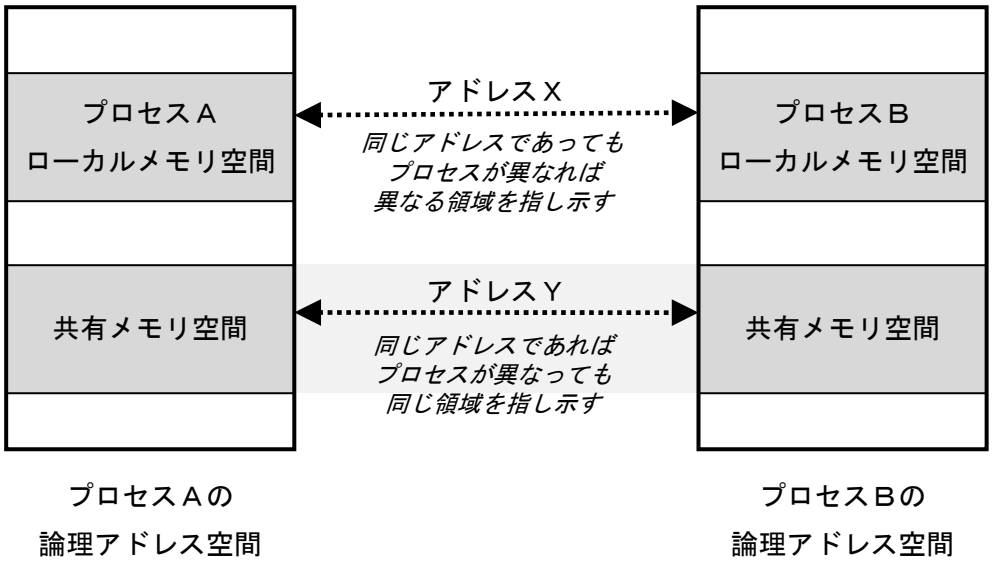
あるプロセスのローカルメモリ空間は、他のプロセスからはアクセスできない。他プロセスのローカルメモリ空間のアドレスにアクセスした場合、そのアドレスに自プロセスのメモリ領域が確保されている場合は自プロセス領域へのアクセスとなり、確保されていない場合はメモリ保護例外が発生する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ローカルメモリ空間に確保した領域のアドレスはプロセス固有のものである。各プロセスが確保した領域のアドレスの値は重複することがあるが、実際にはそれぞれ別の領域を指し示している。このため、ローカルメモリ空間にプロセスＡが確保した領域のアドレスＸを、プロセスＢが使用することはできない。プロセスＡのアドレスＸと、プロセスＢのアドレスＸとはその値が同じでも実際には異なる領域を指し示す。

共有メモリ空間は、どのプロセスからもアクセス可能なメモリ空間である。この空間はプロセス間でデータの受け渡しを行う際に使用できる。

共有メモリ空間に確保した領域のアドレスは各プロセス共通となる。共有メモリ空間にプロセスＡが確保した領域のアドレスＹに対し、プロセスＢからもアドレスＹにアクセスすることで、同じ領域を参照することが可能となる。



[図 3] ローカルメモリ空間と共有メモリ空間

システムメモリ空間は、Standard Extension が内部で使用する特殊なメモリ空間である。これはシステムプログラムやドライバが使用するためのものであり、一般のアプリケーションが使用してはならない。プロセスからシステムメモリ空間の領域にアクセスした場合は、他プロセスのローカルメモリ空間にアクセスした場合と同様にメモリ保護例外が発生する。

2.1.3 プロセス状態とタスク状態

タスクは、その動作の状態に応じてタスク状態を持つ。また、各プロセスのメインタスクのタスク状態をプロセス状態と呼ぶ。

タスク状態は以下５つの基本状態のいずれかとなる。これらのタスク状態はT-Kernelのタスク状態に準じている。ただしタスクを強制待ち状態(SUSPENDED)にすることはできない。また休止状態(DORMANT)はサブタスクのみ遷移可能である。メインタスクを休止状態にすることはできない。起動された状態のタスクを休止状態にすることはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

(1) 実行状態 (RUNNING)

タスクが実行中であることを示す。実行状態のタスクは常に１個以下となる。

(2) 実行可能状態 (READY)

タスクを実行する準備は整っているが、より優先順位の高いタスクが実行中であるため、そのタスクを実行できないことを示す。

実行状態のタスクが実行可能状態または待ち状態に遷移すると、実行可能状態のタスクの中で優先順位が最高のものが新たに実行状態となる。

実行可能状態のタスクがCPU資源を割り当てられて実行状態となることをディスパッチと呼ぶ。また、実行状態のタスクがCPU資源を解放し、実行可能状態に遷移することをプリエンプトと呼ぶ。

(3) 待ち状態 (WAITING)

自タスクの実行を中断するシステムコールを呼び出したことにより、タスクが一時的に実行を中断していることを示す。

(4) 休止状態 (DORMANT)

タスクがまだ起動していない、あるいは実行を終了したことを示す。

タスクが休止状態である場合、タスクの実行状態に関する情報は保存されていない。タスクを起動して休止状態から実行可能状態に遷移した場合、タスクはタスク開始アドレスから実行を開始する。

休止状態はサブタスクのみ遷移可能である。

(5) 未登録状態 (NON-EXISTENT)

タスクがまだ生成されていない、あるいはタスクが削除されたことを示す。

未登録状態は仮想的な状態であり、未登録状態のタスクは、実際はシステムに登録されていない。

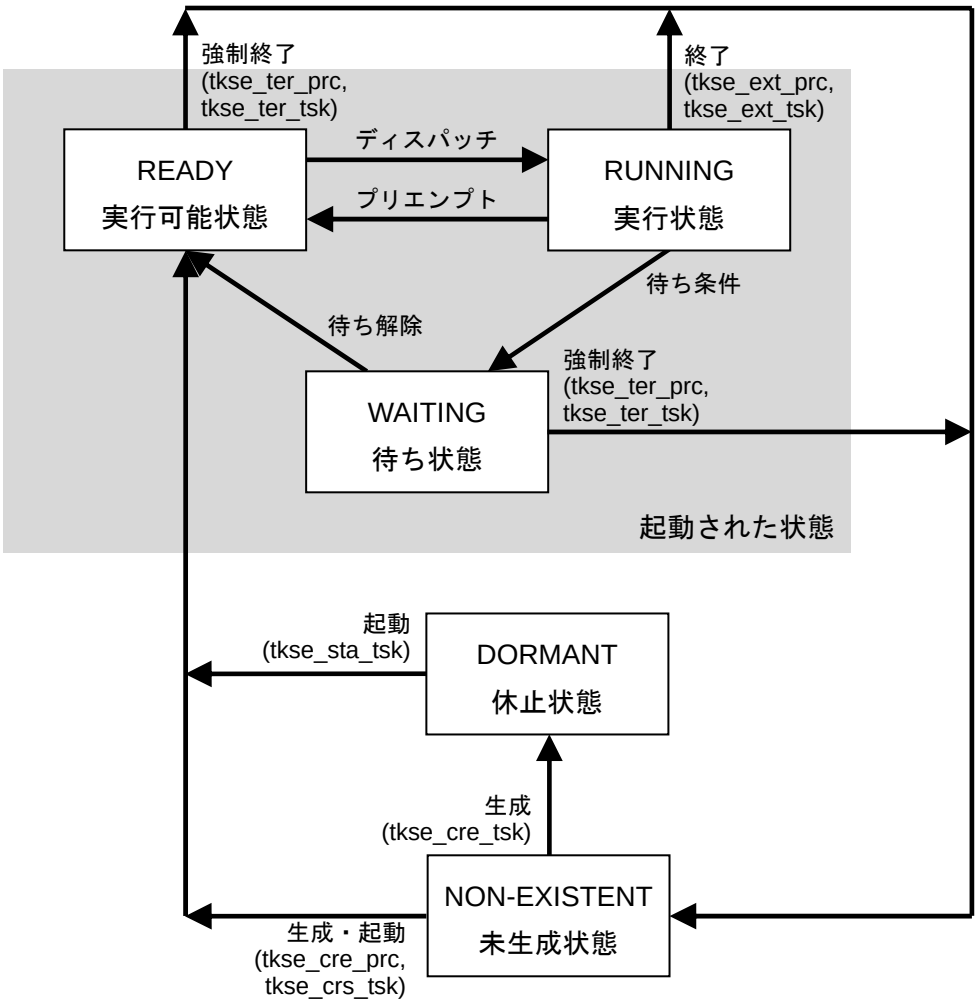
一般的な実装におけるタスクの状態遷移を以下に示す。実装によっては、この図にない状態遷移を行う場合や、いずれの状態にも分類されない過渡的な状態が存在する場合がある。

実行可能状態に移行したタスクが、現在実行中のタスクよりも高い優先順位を持つ場合、実行可能状態への移行と同時にディスパッチが起こり、即座に実行状態へ移行する場合がある。この場合、それまで実行状態であったタスクは、新たに実行状態へ移行したタスクにプリエンプトされたという。また、システムコールの機能説明などで「実行可能状態に移行」すると記述されている場合でも、タスクの優先順位によっては即座に実行状態に移行する場合もある。

タスクの起動とは、休止状態のタスクを実行可能状態に移行させることをいう。このことから、休止状態と未登録状態以外の状態を総称して、起動された状態と呼ぶことがある。タスクの終了とは、起動された状態のタスクを未生成状態に移行させることをいう。

タスクの待ち解除とは、タスクを待ち状態から実行可能状態に移行させることをいう。また、待ち状態を解除する要因のことを待ち解除要因と呼ぶ。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.



[図 4] タスク状態遷移図

2.1.4 プロセス／タスクの優先度とスケジューリング

タスクはそれぞれ独立したタスク優先度を持つ。プロセスのメインタスクのタスク優先度を、プロセス優先度と呼ぶ。またサブタスクもそれぞれ優先度を持っており、これらのサブタスク優先度はプロセス優先度と異なる値を設定可能である。優先度の設定はプロセスおよびサブタスク生成時に行う。また、タスク動作中に優先度を動的に変更することも可能である。

優先度は0～255（0が最高優先度）の範囲の値をとる。優先度の値に応じて、各タスクは3つの優先度グループに分類され、それぞれ異なったスケジューリングが行われる。Standard Extensionのスケジューリングは基本的には、絶対優先度スケジューリングとラウンドロビンスケジューリングの二つがある。

絶対優先度スケジューリングは、タスクの優先度が高いほど、タスクの優先順位が高くなる。よって、あるタスクが実行状態のときに、そのタスクより優先度の低いタスクが実行状態となることはない。このスケジューリングは、T-Kernelのスケジューリングと基本的に同一である。

ラウンドロビンスケジューリングは、タスクの優先度に関係なく、順番にタスクを実行状態としていく。タスクの優先度は、相対的なスケジューリングの頻度を示す。より具体的には、タスクに割り当てられる実行時間（実行状態でいられる時間）が、優先度が高いほど多くなる。割り当てられた実行時間が過ぎると、優先順位は最も

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

低くなり、他のタスクが実行状態となる。つまり優先度が低いタスクでも必ず実行される。
各タスクは、優先度の値により、以下の3つの優先度グループに分類される。

A. 絶対優先度グループ（優先度: 0～127）
グループ内で、タスク優先度に基づいた絶対優先度スケジューリングが行われる（0が最高優先度）。
ただし、タスクの優先度が同一の場合はラウンドロビン方式で一定時間ごとに平等にスケジューリングされる。
優先度の高いタスクが実行状態／実行可能状態である場合は、優先度の低いタスクが動作することはない。

B. ラウンドロビングループ１（優先度: 128～191）
グループ内で、ラウンドロビン方式によるスケジューリングが行われる（128が最高優先度）。
このグループは、絶対優先度グループより優先順位が低い。よって、絶対優先度グループに実行状態／実行可能状態のタスクが存在すれば、このグループのタスクが実行されることはない。絶対優先度グループに実行状態／実行可能状態のタスクが存在しなければ、優先度が低いタスクでも必ず実行されることが保証される。

C. ラウンドロビングループ２（優先度: 192～255）
グループ内で、ラウンドロビン方式によるスケジューリングが行われる（192が最高優先度）。
このグループは、他のグループ（絶対優先度グループおよびラウンドロビングループ１）より優先順位が低い。よって、他のグループに実行状態／実行可能状態のタスクが存在すれば、このグループのタスクが実行されることはない。他のグループに実行状態／実行可能状態のタスクが存在しなければ、優先度が低いタスクでも必ず実行されることが保証される。

実際のスケジューリングは以下のように行われる。

1. 絶対優先度グループに属する実行可能状態のタスクがあれば、その中の最高優先度のタスクを実行状態とし、実行する。なければ、2. へ進む。
最高優先度のタスクが複数存在した場合は、ラウンドロビン方式で一定時間ごとに平等にスケジューリングされる。

2. ラウンドロビングループ１に属する実行可能状態のタスクがあれば、その中の相対的な優先順位に従って、選択されたタスクを実行状態とし、実行する(最高優先度とは限らない)。なければ、3. へ進む。

3. ラウンドロビングループ２に属する実行可能状態のタスクがあれば、その中の相対的な優先順位に従って、選択されたタスクの状態を実行状態に変更し、タスクを実行する（このため、実行されているタスクが最高優先度であるとは限らない）。実行すべきタスクが存在しなければ、スケジューリングを始めからやり直す。

2.1.5 プロセスの実行環境

プロセスは、実行環境として以下の情報を保持する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

・自プロセス、親プロセス、子プロセスのプロセス ID
・プロセス／タスクの優先度
・現在の作業ファイル（標準ファイル管理）
・オープン中のファイル（標準ファイル管理、標準入出力）
・メッセージキュー（プロセス間メッセージ）

プロセス生成直後の実行環境は以下のように設定される。

・自プロセスのプロセス ID　・・・　生成時に割り当てられた ID
・親プロセスのプロセス ID　・・・　自プロセスを生成したプロセスの ID
・子プロセスのプロセス ID　・・・　なし
・プロセス／タスクの優先度　・・・　生成時に指定した優先度
・現在の作業ファイル　・・・　生成時点での親プロセスの作業ファイル
・オープン中のファイル　・・・　なし
・メッセージキュー　・・・　空

また、セマフォなどのカーネルオブジェクトは、生成時の属性指定(TA_DELEXIT)により、オブジェクトを生成したプロセスに関連付けることが可能である。プロセスに関連付けられたオブジェクトは、該当するプロセスが終了した時点で自動的に削除される。

2.1.6 ユーザプロセスとシステムプロセス

プロセスは、ユーザプロセスとシステムプロセスの２種類が存在する。ユーザプロセスまたはシステムプロセスのどちらでプロセスを生成するかの指定は、プロセス生成時の属性指定で行う。

ユーザプロセスは Standard Extension のすべての機能を使用できる。システムプロセスは、ユーザプロセスの機能に加えて T-Kernel のシステムコール(tk_xxx_yyy など)を直接使用できる。

システムプロセスはデバッガや上位システムとの連携など、システムに近い用途での利用を想定したものである。一般的なアプリケーションは原則としてユーザプロセスとしなければならない。

2.1.7 プロセスの生成

プロセスの生成は、プロセスの実行プログラムファイル、およびプロセス生成メッセージを指定して、システムコール tkse_cre_prc を発行することで行われる。

プロセス生成メッセージは、プロセス生成時に親プロセスから子プロセスに渡されるメッセージである。プロセス生成メッセージの構造は、プロセス間メッセージで使われる通常のメッセージの構造と同一である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
typedef struct {
        W msg_type;          /* メッセージタイプ */
        W msg_size;          /* メッセージサイズ(バイト数) */
        UB msg_body[n];      /* メッセージ本体(msg_size バイト) */
} MESSAGE;
```

プロセス生成に成功すると、プロセスのメインタスクが起動する。このとき、メインタスク関数は引数としてプロセス生成メッセージを受け取る。

メッセージを受け取る形式として、メッセージデータを直接受け取る形式と、メッセージデータを空白で区切られた文字列とみなし、個々の要素に分解して受け取る形式の２種類を選択できる。ユーザの用途に応じて、メインタスク関数の名称の定義を以下から選択する。ただし両方の名称を同時に定義することはできない。

(1)形式１

```
W MAIN (MESSAGE *msg)
/* MESSAGE *msg; メッセージへのポインタ */
{
        /* プログラムの実行コード */
        return 終了コード;
}
```

メインタスク関数の名称を MAIN として定義した場合､関数引数としてプロセス生成メッセージ msg を直接受け取る。このとき、メッセージタイプ msg_type の値に制限はない。

(2)形式２

```
W main (W ac, TC **argv)
/* W ac; 文字列項目数 */
/* TC **argv; 文字列項目のポインタ配列へのポインタ */
{
        /* プログラムの実行コード */
        return 終了コード;
}
```

メインタスク関数の名称を main として定義した場合、プロセス生成メッセージのメッセージデータ msg_body は、空白文字 TK_KSP で区切られた、TNULL で終了する 1 つの TRON コード文字列と見なされる。このとき、メインタスク関数の引数 ac には空白文字で区切られた項目の個数が、argv には各項目の文字列へのポインタがポインタ配列として渡される。

msg_body が TNULL で終了していない場合は､msg_body の終端の文字を TNULL に置き換えて引数の解析処理を行う。このとき msg_body の終端の文字は失われる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

形式２を使用する場合はメッセージタイプ msg_type = 0 でなくてはならない。msg_type≠0 を指定した場合は msg_body の内容に関わらず常に ac = 0, *argv = NULL となり、メッセージを受け取ることはできない。

メインタスク関数 MAIN または main から処理が戻るとプロセスは終了する。これは tkse_ext_prc によるプロセス終了と同等となる。

2.1.8 T-Kernel プログラムとの連携

Standard Extension 上で動作するアプリケーションは、T-Kernel プログラムと連携して動作することが可能である。アプリケーションからアクセス可能な T-Kernel プログラムは、主に以下の２種類である。

・デバイスドライバ
　システムに接続されている各種デバイスを制御する。
　Standard Extension のデバイス管理機能を使用してアクセスする。
・サブシステム
　システムに機能を追加するために、各種ミドルウェアなどで使用される。
　各サブシステムが提供する拡張 SVC を使用してアクセスする。

これらの T-Kernel プログラムを、まとめてシステムプログラムと呼ぶ。システムプログラムは T-Kernel と同じシステムメモリ空間上で動作する。

システムプログラムは T-Kernel に直接リンクすることでメモリ空間に配置される。また、アプリケーションから動的にロード・アンロードすることも可能である。

システムプログラムの実行ファイルを指定して tkse_lod_spg を発行することにより、アプリケーションからシステムプログラムをロードすることができる。このときロードする領域は動的に確保されるため、システムプログラムは再配置可能な形式で作成しなくてはならない。ただし、実行ファイルに格納されているシステムプログラムの配置アドレスが OS 管理外の論理アドレスである場合は、実行ファイルの配置情報に従って固定アドレスにロードされる。

ロードしたシステムプログラムは、以下の形式の main 関数から実行が開始される。プロセス生成とは異なり MAIN 関数を使用することはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
W main (W ac, TC **argv)
/* W ac; 文字列項目数 */
/* TC **argv; 文字列項目のポインタ配列へのポインタ */
{
        if (ac >= 0) {
            /* プログラムのロード処理 */
        } else {
            /* プログラムのアンロード処理 */
        }
        return 終了コード;
}
```

tkse_lod_spg でシステムプログラムをロードした際の引数 arg を、空白文字 TK_KSP で区切られた、TNULL で終了する 1 つの TRON コード文字列と見なす。main 関数の引数 ac には空白文字で区切られた項目の個数が、argv には各項目の文字列へのポインタがポインタ配列として渡される。ロード時は常に ac ≧ 0 となる。

tkse_unl_spg で、ロードしたシステムプログラムをアンロードする。このときもロード時と同様に main 関数が呼び出される。ただしアンロード時は ac ＜ 0 となるため、ac の値を判定してロード処理・アンロード処理をそれぞれ行う。

main 関数は tkse_lod_spg を呼び出したタスクの準タスク部として実行される。main 関数では T-Kernel の API を使用可能であるため、サブシステムの場合はここでサブシステム定義・抹消を、デバイスドライバの場合はデバイスの登録・抹消を行う。

呼び出し元タスクの実行状態に影響を与えるため、main 関数内でタスク終了等の自タスク状態の変更を行ってはならない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

2.2 同期･通信

2.2.1 プロセス間の同期・通信

プロセス間で通信を行うため、Standard Extension は以下の機能を提供する。

(1) プロセス間メッセージ

プロセス間メッセージ機能は、メッセージと呼ばれるデータ構造を送信側プロセスから受信側プロセスに送ることで、１対１のプロセス間通信を実現する機能である。また、プロセス間の同期にも利用可能である。

メッセージ送信処理(tkse_snd_msg)により送信されたメッセージは、受信側プロセスのメッセージキューに格納される。メッセージキューは各プロセスに固有のものであり、プロセス生成時に自動的に生成･初期化される。受信側プロセスはメッセージ受信処理(tkse_rcv_msg)を行うことで、自プロセスのメッセージキューに格納されたメッセージを取り出す。また、受信側プロセスにメッセージハンドラを定義することで、非同期のメッセージ受信を行うことも可能である。この場合、受信側プロセスがメッセージを受け取ったタイミングで、メインタスクの処理に割り込む形でメッセージハンドラが起動する。

プロセス間メッセージは、プロセス対プロセスの通信だけでなく、システムがプロセスに情報を伝える手段としても用いられる。例として、子プロセスが終了した場合、システムから親プロセスに子プロセス終了メッセージが送信される。このような、システムが送信するメッセージをシステムメッセージと呼ぶ。

(2) グローバル名

グローバル名機能は、グローバル名と呼ばれる任意の名前を付与した 4byte データを、複数プロセス間で共有する機能である。

プロセスはそれぞれ独立したローカルメモリ空間を持つため、ローカルメモリ空間内の大域変数などを利用して、複数のプロセス間でデータを共有することはできない。データの共有は共有メモリやメッセージバッファ等を利用するが、これらの機能を使用するためには、共有メモリの領域のアドレス、あるいはメッセージバッファ等のオブジェクト ID をまず共有しなければならない。このような、アドレスや ID を共有するための機能がグローバル名機能である。

グローバル名機能の用途はアドレスや ID の共有を想定しているが、4byte データであればどのようなデータでも共有可能である。

(3) 共有メモリ

共有メモリによるプロセス間通信は、前述の共有メモリ空間を用いてデータを受け渡す方法である。

大量のデータを複数のプロセスからアクセスする場合に使用する。データのコピーを行わないため高速なデータの受け渡しが可能であるが、アクセス保護が行われない点や、同期や排他制御を行うためには他の機能を併用しなければならない点を考慮する必要がある。

プロセス間の同期はプロセス間メッセージ機能を使用する。タスク単位の、より細かな同期が必要な場合はタスク間同期・通信機能を用いる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

2.2.2 タスク間の同期･通信

(1) タスク間同期・通信機能

タスク間の同期･通信は、同期･通信のために用意されたオブジェクトを使用して実現する。また、これらのオブジェクトが提供する機能をタスク間同期・通信機能と呼ぶ。

タスク間同期･通信機能として以下のオブジェクトを使用可能である。

・セマフォ
・ミューテックス
・イベントフラグ
・メールボックス
・メッセージバッファ
・ランデブポート

これらのタスク間同期･通信機能を使用するために、まず該当する機能のオブジェクトを生成する。生成したオブジェクトには固有のオブジェクト ID が割り付けられる。このオブジェクト ID を指定して、タスク間の同期や通信を行う。

タスク間同期･通信は、同一プロセス内のタスク間だけでなく、異なるプロセスに属するタスク間で行うことも可能である。ただし、メールボックスは同一プロセス内のタスク間でのみ使用可能である。

Standard Extension のタスク間同期･通信機能の仕様は、T-Kernel のタスク間同期･通信機能の仕様に準じる。ただしオブジェクト ID は Standard Extension が独自に管理しているため、Standard Extension で生成したオブジェクトのオブジェクト ID を T-Kernel でそのまま利用することはできない。また T-Kernel で生成したオブジェクトのオブジェクト ID を Standard Extension で利用することもできない。また、オブジェクト生成時の属性指定に一部制約がある（仕様の詳細は各システムコールの説明を参照）。

メインタスクがタスク間同期・通信機能を使用して待ち状態となっている間に、プロセスのメッセージハンドラが割り込んだ場合は、タスクの待ち状態は解除されシステムコールはエラーE_DISWAI を返す。

(2) タスク付属同期機能

タスク間同期・通信機能を用いず、他タスクの状態を直接操作することでタスク間の同期を行う事も可能である。他タスクの状態を制御して同期を行う機能をタスク付属同期機能と呼ぶ。

Standard Extension で使用可能なタスク付属同期機能は、タスクの起床と起床待ち、およびその解除である。

タスク付属同期機能は同一プロセス内のタスクにのみ使用可能である。他プロセスのタスクのタスク状態を操作することはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

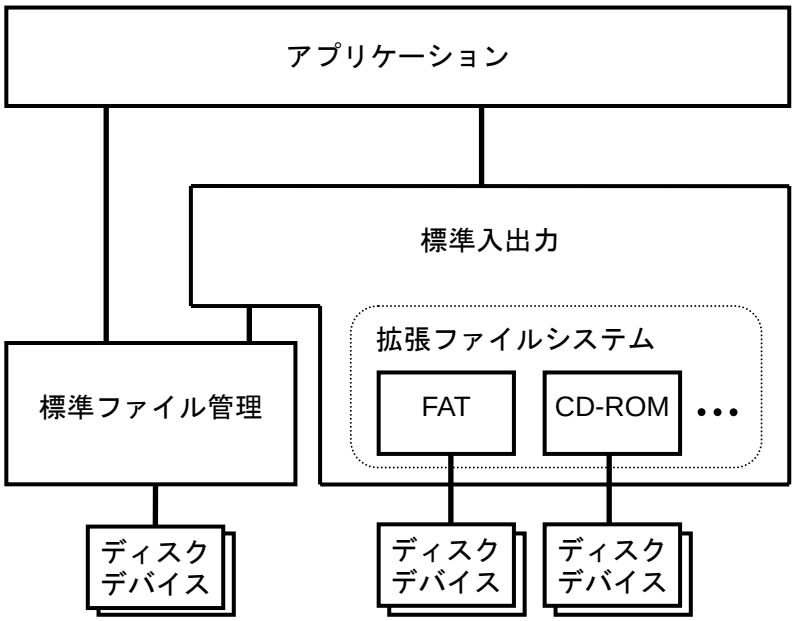
2.3 標準ファイル管理と標準入出力

2.3.1 Standard Extension のファイル管理

Standard Extension は、T-Kernel に登録されたディスクデバイスをファイルシステムとして扱うためのファイル管理機能を持つ。

ファイル管理機能は、T-Kernel 標準ファイルシステム（以下、標準ファイルシステム）を直接操作するための標準ファイル管理機能、および標準ファイルシステムを含む様々なファイルシステムを統一的に扱うための標準入出力機能からなる。

標準入出力は標準ファイルシステムだけではなく他のフォーマットのファイルシステムも扱うことができる。これらの他フォーマットのファイルシステムを拡張ファイルシステムと呼ぶ。現バージョンの仕様では FAT12/16/32 ファイルシステム、および CD-ROM(ISO9660 Level1)ファイルシステムに対応する。また、これら以外のファイルシステムを拡張ファイルシステムとして標準入出力に組み込むことも可能である。



[図 5] ファイル管理の位置付け

ディスクデバイスをファイルシステムとして使用するため、最初にファイルシステムの接続を行う。接続したファイルシステムは一意の接続名を持ち、この接続名を用いてファイルシステム上に存在するファイルの操作を行う。標準ファイル管理、および標準入出力は、複数の異なるファイルシステムを同時に接続することができる。実行ファイルからのプロセス・システムプログラム起動、およびページファイルを利用した仮想記憶管理を行うためには、あらかじめファイルシステムを接続しておかなければならない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

2.3.2 標準ファイル管理

標準ファイル管理は、ハイパーテキスト型のネットワーク構造を持つ標準ファイルシステムを直接扱うための機能である。標準ファイルシステム独自の機能である実身／仮身やファイルレコードの操作を行う場合は、標準ファイル管理機能を使用する。

2.3.3 標準入出力

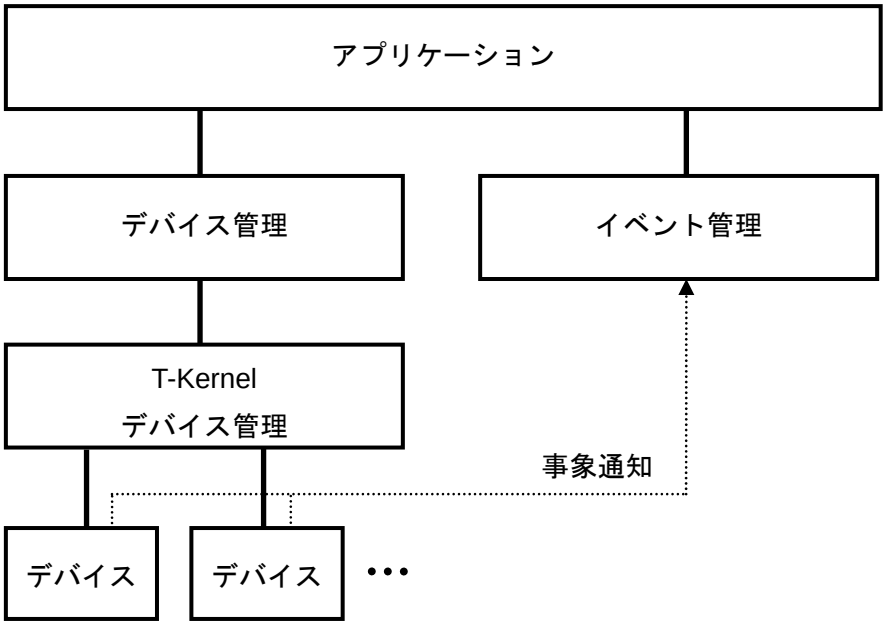
標準入出力は、ファイルシステム毎の仕様の差異を意識することなく、アプリケーションからのファイルアクセスを共通のシステムコールで実現するための機能である。ただし、ファイル名長や最大ファイルサイズなどの元々のファイルシステムに起因する制限事項は、標準入出力でも同様に制限される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

2.4 デバイス管理とイベント管理

2.4.1 T-Kernel デバイスのアクセスと事象通知

Standard Extension は、T-Kernel に登録されたデバイスにアクセスするためのデバイス管理機能、およびデバイスから送信された事象通知をアプリケーションが受け取るためのイベント管理機能を提供する。



[図 6] デバイス管理とイベント管理の位置付け

2.4.2 デバイス管理

デバイス管理は、T-Kernel のデバイス管理機能を Standard Extension を通して利用するための機能である。実際のデバイスの操作、およびデバイスの管理は T-Kernel が行う。

デバイスの登録は T-Kernel からのみ実行可能である。Standard Extension からデバイスを登録、および登録解除することはできない。

2.4.3 イベント管理

イベント管理は、各種デバイスから非同期に発生する事象通知をアプリケーションが受け取るための機能である。デバイスの事象通知はイベントと呼ばれるデータ構造に変換され、イベント管理内部のイベントキューに順次格納される。イベントキューはシステム全体で１つしか存在しないため、複数のプロセスから同時にイベント管理機能を使用することはできない。

アプリケーションはイベントの取得を行うことで、イベントキューに格納されたイベントを取り出すことがで

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

きる。また、イベントをメッセージとして受け取ることもできる。

　イベント管理の主な目的はインタラクティブなヒューマンインタフェースを実現することである。このためイベント管理は、キーボードやポインティングデバイスといったデバイスからの事象通知を、イベントとしてアプリケーションに通知することを想定して設計されている。ただし、デバイスイベント、拡張デバイスイベント、アプリケーションイベントなどを使用して、その他のデバイスからの事象通知をアプリケーションに送信する目的にも使用することができる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

第3章　T-Kernel Standard Extension 共通仕様規定

3.1　データ型

3.1.1　基本的なデータ型

```
typedef     char            B;          /* 符号付き 8 ビット整数 */
typedef     short           H;          /* 符号付き 16 ビット整数 */
typedef     int             W;          /* 符号付き 32 ビット整数 */
typedef     unsigned char   UB;         /* 符号無し 8 ビット整数 */
typedef     unsigned short  UH;         /* 符号無し 16 ビット整数 */
typedef     unsigned int    UW;         /* 符号無し 32 ビット整数 */

typedef     char            VB;         /* 型が一定しない 8 ビットのデータ */
typedef     short           VH;         /* 型が一定しない 16 ビットのデータ */
typedef     int             VW;         /* 型が一定しない 32 ビットのデータ */
typedef     void            *VP;        /* 型が一定しないデータへのポインタ */

typedef     volatile B      _B;         /* volatile 宣言付 */
typedef     volatile H      _H;
typedef     volatile W      _W;
typedef     volatile UB     _UB;
typedef     volatile UH     _UH;
typedef     volatile UW     _UW;

typedef     int             INT;        /* プロセッサのビット幅の符号付き整数 */
typedef     unsigned int    UINT;       /* プロセッサのビット幅の符号無し整数 */

typedef     INT             ID;         /* ID 一般 */
typedef     INT             MSEC;       /* 時間一般(ミリ秒) */
typedef     void            (*FP)();    /* 関数アドレス一般 */
typedef     INT             (*FUNCP)(); /* 関数アドレス一般 */

#define     LOCAL           static      /* ローカルシンボル定義 */
#define     EXPORT                      /* グローバルシンボル定義 */
#define     IMPORT          extern      /* グローバルシンボル参照 */

/*
 * ブール値
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
 * TRUE = 1 と定義するが、0 以外はすべて真(TRUE)である。
 * したがって、bool == TRUE の様な判定をしてはいけない。
 * bool != FALSE の様に判定すること。
 */
typedef     INT             BOOL;
#define     TRUE            1           /* 真 */
#define     FALSE           0           /* 偽 */

/*
 * TRON コード
 */
typedef     UH              TC;         /* TRON コード */
#define     TNULL           ((TC)0)     /* TRON コード文字列の終端 */
```

※VB,VH,VW と B,H,W との違いは、前者はビット数のみが分かっており、データ型の中身が分からないものを表すのに対して、後者は整数を表すことがはっきりしているという点である。
※プロセッサのビット幅は 32 ビット以上に限定する。したがって、INT と UINT は必ず 32 ビット以上の幅がある。
※BOOL は、TRUE を 1 と定義するが、0 以外はすべて真である。したがって、ブール値==TRUE の様な判定を行ってはいけない。ブール値!=FALSE の様に判定する。

【補足事項】
明らかに負の数にならないパラメータも、原則として符号付き整数(INT)のデータ型を使用する。これは、整数はできるだけ符号付きの数として扱うという TRON 全般のルールに基づいたものである。また、タイムアウト(TMO tmout)のパラメータは、これが符号付きの整数であることを利用し、TMO_FEVR(=-1)を特殊な意味に使っている。符号無しのデータ型を持つパラメータは、ビットパターンとして扱われるもの(オブジェクト属性やイベントフラグなど)である。

3.1.2 意味が定義されているデータ型

パラメータの意味を明確にするため、出現頻度の高いデータ型や特殊な意味を持つデータ型に対して、以下のような名称を使用する

```
typedef     INT             FN;         /* 機能コード */
typedef     INT             RNO;        /* ランデブ番号 */
typedef     UINT            ATR;        /* オブジェクト/ハンドラ属性 */
typedef     INT             ER;         /* エラーコード */
typedef     INT             PRI;        /* 優先度 */
typedef     INT             TMO;        /* タイムアウト指定 */
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
typedef     UINT                RELTIM;         /* 相対時間 */
typedef     struct systim {                     /* システム時刻 */
                W           hi;                 /* 上位 32 ビット */
                UW          lo;                 /* 下位 32 ビット */
} SYSTIM;

/*
 * 共通定数
 */
#define     NULL                0               /* 無効ポインタ */
#define     TA_NULL             0               /* 特別な属性を指定しない */
#define     TMO_POL             0               /* ポーリング */
#define     TMO_FEVR            (-1)            /* 永久待ち */
```

※複数のデータ型を複合したデータ型の場合は、その内の最も主となるデータ型で代表する。例えば、tkse_cre_prc の戻値はプロセス ID かエラーコードであるが、主となるのはプロセス ID なのでデータ型は ID となる。

3.2 エラーコード

3.2.1 概要

システムコールの戻値は原則として符号付きの整数で、エラーが発生した場合には負の値のエラーコード、処理を正常に終了した場合は E_OK(＝0)または正の値とする。正常終了した場合の戻値の意味はシステムコール毎に規定する。この原則の例外として、呼び出しても戻値を返さないシステムコールがある。

戻値を返さないシステムコールは、C 言語 API では戻値を持たないもの(すなわち void 型の関数)として宣言する。

エラーコードは、メインエラーコードとサブエラーコードで構成される。エラーコードの下位 16 ビットがサブエラーコード、残りの上位ビットがメインエラーコードとなる。メインエラーコードは、検出の必要性や発生状況などにより、エラークラスに分類される。

```
#define     MERCD(er)           ( (ER)(er) >> 16 )              /* メインエラーコード */
#define     SERCD(er)           ( (H)(er) )                     /* サブエラーコード */
#define     ERCD(mer, ser)      ( (ER)(mer) << 16 | (ER)(UH)(ser) )
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

3.2.2 エラーコード一覧

以下に Standard Extension のエラーコードを示す。メインエラーコードのうち 0～-255 までは T-Kernel 互換エラーコードであり、エラーの意味は T-Kernel と同一である。-256 以降は Standard Extension 独自のエラーコードである。

エラークラスとして定義されていない範囲のエラーコードは、将来の拡張用に予約されている。

――― 正常終了のエラークラス (0) ―――――――――――――――――――――――――――――――――――――――

E_OK　　0　　正常終了

――― 内部エラークラス (-5～-8) ―――――――――――――――――――――――――――――――――――――――

E_SYS　　ERCD(-5, 0)　　システムエラー

原因不明のエラーであり、システム全体に影響するエラーである。

E_NOCOP　　ERCD(-6, 0)　　コプロセッサ使用不可

現在動作中のハードウェアに指定のコプロセッサが搭載されていない。または、コプロセッサの動作異常を検出した。

――― 未サポートエラークラス (-9～-16) ―――――――――――――――――――――――――――――――――――

E_NOSPT　　ERCD(-9, 0)　　未サポート機能

システムコールの一部の機能がサポートされていない場合に、その機能を指定すると、E_RSATR または E_NOSPT のエラーを発生する。E_RSATR に該当しない場合には、E_NOSPT のエラーとなる。

E_RSFN　　ERCD(-10, 0)　　予約機能コード番号

予約機能コード(未定義の機能コード)を指定してシステムコールを実行しようとした場合に、このエラーが発生する。未定義の拡張 SVC ハンドラを実行しようとした場合(機能コードが正の場合)にも、このエラーが発生する。

E_RSATR　　ERCD(-11, 0)　　予約属性

未定義やサポートしていないオブジェクト属性を指定した場合に発生する。

システム依存の適応化を行う場合、このエラーのチェックは省略されることがある。

――― パラメータエラークラス (-17～-24) ―――――――――――――――――――――――――――――――――――

E_PAR　　ERCD(-17, 0)　　パラメータエラー

システム依存の適応化を行う場合、このエラーのチェックは省略されることがある。

E_ID　　ERCD (-18, 0)　　不正 ID 番号

E_ID は ID 番号を持つオブジェクトに対してのみ発生するエラーである。

割込み定義番号などの範囲外や予約番号といった静的なエラーが検出された場合には、E_PAR のエラーが発生

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

する。

――― 呼出コンテキストエラークラス (-25～-32) ―――――――――――――――――――――――――――――――

E_CTX　　ERCD(-25, 0)　　コンテキストエラー

このシステムコールを発行できるコンテキスト(タスク部/タスク独立部の区別やハンドラ実行状態)にはないということを示すエラーである。

自タスクを待ち状態にするシステムコールをタスク独立部から発行した場合のように、システムコールの発行コンテキストに関して意味的な間違いのある場合には、必ずこのエラーが発生する。また、それ以外のシステムコールであっても、実装の制約のため、あるコンテキスト(割込みハンドラなど)からそのシステムコールを発行できない場合に、このエラーが発生する。

E_MACV　　ERCD(-26, 0)　　メモリアクセス不能、メモリアクセス権違反

エラーの検出は実装依存である。

E_OACV　　ERCD(-27, 0)　　オブジェクトアクセス権違反

ユーザタスクがシステムオブジェクトを操作した場合に発生する。

システムオブジェクトの定義およびエラーの検出は実装依存である。

E_ILUSE　　ERCD(-28, 0)　　システムコール不正使用

――― 資源不足エラークラス (-33～-40) ―――――――――――――――――――――――――――――――

E_NOMEM　　ERCD(-33, 0)　　メモリ不足

オブジェクト管理ブロック領域、ユーザスタック領域、メモリプール領域、メッセージバッファ領域などを獲得する時のメモリ不足 (no memory)

E_LIMIT　　ERCD(-34, 0)　　システムの制限を超過

オブジェクト数の上限を超えてオブジェクトを生成しようとした場合など。

――― オブジェクト状態エラークラス (-41～-48) ―――――――――――――――――――――――――――

E_OBJ　　ERCD(-41, 0)　　オブジェクトの状態が不正

E_NOEXS　　ERCD(-42, 0)　　オブジェクトが存在していない

E_QOVR　　ERCD(-43, 0)　　キューイングまたはネストのオーバーフロー

――― 待ち解除エラークラス (-49～-56) ―――――――――――――――――――――――――――――――

E_RLWAI　　ERCD(-49, 0)　　待ち状態強制解除

E_TMOUT　　ERCD(-50, 0)　　ポーリング失敗またはタイムアウト

E_DLT　　ERCD(-51, 0)　　待ちオブジェクトが削除された

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_DISWAI　ERCD(-52, 0)　待ち禁止による待ち解除

――― デバイスエラークラス (-57～-64) (T-Kernel/SM) ―――

E_IO　ERCD(-57, 0)　入出力エラー

※ E_IO のサブエラーコードには、デバイスごとにエラー状態等を示す値が定義される場合がある。

E_NOMDA　ERCD(-58, 0)　メディアがない

――― 各種状態エラークラス (-65～-72) (T-Kernel/SM) ―――

E_BUSY　ERCD(-65, 0)　ビジー状態

E_ABORT　ERCD(-66, 0)　中止した

E_RONLY　ERCD(-67, 0)　書込み禁止

――― メモリ管理エラークラス (-257～-260) (Standard Extension) ―――

E_SYSMEM　ERCD(-257, 0)　システムメモリ領域不足

Standard Extension が内部で使用するメモリ領域が不足した場合に発生する。

――― ファイル管理エラークラス (-261～-280) (Standard Extension) ―――

E_FNAME　ERCD(-261, 0)　不正パス名、不正ファイル名

E_FD　ERCD(-262, 0)　不正ファイルディスクリプタ

E_FACV　ERCD(-263, 0)　ファイルのアクセス権違反

E_PERM　ERCD(-264, 0)　削除不可ファイル

E_PWD　ERCD(-265, 0)　不正パスワード

Standard Extension では使用しない。

E_ENDR　ERCD(-266, 0)　終端レコードに達した

E_REC　ERCD(-267, 0)　不正レコードタイプ

E_NOLNK　ERCD(-268, 0)　リンクファイルではない

E_LOCK　ERCD(-269, 0)　レコードはロックされている

E_XFS　ERCD(-270, 0)　異なるファイルシステムに属している

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_NOFS	ERCD(-271, 0)	ファイルシステム未接続
E_NODSK	ERCD(-272, 0)	ディスクの領域不足
E_ILFMT	ERCD(-273, 0)	ディスクのフォーマットが不正である
E_SEIO	ERCD(-274, 0)	標準入出力のエラー

――― デバイス管理エラークラス (-281～-290) (Standard Extension) ―――――――――――――――――――――

E_NODEV	ERCD(-281, 0)	デバイスが存在しない
E_ERDEV	ERCD(-282, 0)	デバイス状態が異常である

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

第4章　T-Kernel Standard Extension の機能

4.1　メモリ管理

4.1.1　概要

Standard Extension のメモリ管理機能は、データを格納するための各種メモリ空間を管理する機能である。管理するメモリ空間は、ローカルメモリ空間、共有メモリ空間、システムメモリ空間の３種類があり、それぞれの空間に対して指定した大きさのメモリ領域を割り当て／解放する機能を提供する。

メモリ管理機能のメモリ領域割り当てはブロック単位となる。ブロック単位のメモリ領域をメモリブロックと呼ぶ。ブロックのサイズは実装依存であり、MMU などのハードウェア仕様に依存した値となる。通常は、ブロック単位より細かな単位でメモリ領域を管理するために、メモリ管理機能のシステムコールを直接使用するのではなく、メモリ管理ライブラリを使用する。ただしライブラリの機能で不十分な場合は、アプリケーションから直接メモリ管理機能のシステムコールを利用することも可能である。

システムコールで割り当てたメモリブロックは連続した論理アドレスを持ち、そのメモリブロックの先頭の論理アドレスがシステムコールの戻値としてアプリケーションに返される。一度割り当てたメモリブロックの論理アドレスは変更されることはないため、得られたアドレスを使用してメモリブロックに直接アクセスすることができる。割り当てたメモリブロックへのデータの書き込み／読み込みは自由にできるが、原則としてプログラムを実行することはできない。

メモリブロックに対する排他制御の機能はないため、必要な場合はセマフォなどを利用してアプリケーション側で排他制御を行う。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.1.2 システムコール

メモリブロックの割り当て

tkse_get_mbk

C言語インタフェース

```
ER ercd = tkse_get_mbk(VP *adr, INT nblk, UINT atr);
```

パラメータ

VP	*adr	割り当てたメモリブロックの先頭アドレスを返す領域
INT	nblk	割り当てたメモリブロック数(> 0)
UINT	atr	メモリブロックの属性
		[(M_COMMON ・ M_SYSTEM)] | [M_RESIDENT] | [TA_DELEXIT]
		M_COMMON : 共有メモリ空間の領域を割り当てる
		M_SYSTEM : システムメモリ空間の領域を割り当てる
		M_RESIDENT : 常駐メモリとする
		TA_DELEXIT : プロセス終了時にメモリブロックを削除する

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(adr)のアクセスは許されていない
E_NOMEM	メモリ領域が不足した
E_PAR	パラメータが不正である
E_SYSMEM	システムのメモリ領域が不足した

解説

nblk で指定したブロック数分の連続したメモリ領域を割り当て、その先頭アドレスを *adr に返す。
atr にはメモリブロックの属性を指定する。

・M_COMMON 属性を指定した場合、共有メモリ空間にメモリブロック領域を割り当てる。このメモリブロックはすべてのプロセスからアクセス可能となる。
・M_SYSTEM 属性を指定した場合、システムメモリ空間にメモリブロック領域を割り当てる。このメモリブロックは通常のプロセスからはアクセスできず、システム（OS やデバイスドライバ等）からのみアクセス可能とな

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

る。M_SYSTEM の指定は、アプリケーションからは原則として使用してはいけない。

・M_COMMON, M_SYSTEM のいずれの属性も指定しなければ、ローカルメモリ空間にメモリブロック領域を割り当てる。このメモリブロックは、割り当てたプロセスからのみアクセス可能である。

M_RESIDENT 属性を指定した場合、割り当てたメモリブロックは常駐メモリとなる。常駐メモリはディスクなどにスワップアウトされることなく常に物理メモリ上に存在する。指定がなければ非常駐メモリとなる。

仮想記憶を行っていないシステムの場合、M_RESIDENT 属性の指定は意味を持たない（常駐と同等になる）。

TA_DELEXIT 属性を指定した場合、メモリブロックを割り当てたプロセスが終了すると自動的にメモリブロックを解放する。ただし、ローカルメモリ空間に割り当てたメモリブロックは TA_DELEXIT の指定と関係なく、プロセスの終了時に必ず解放される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メモリブロックの解放

tkse_rel_mbk

C言語インタフェース

```
ER ercd = tkse_rel_mbk(VP adr);
```

パラメータ

VP	adr	解放するメモリブロックのアドレス

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	メモリブロックのアドレスが不正である

解説

adr で示すメモリブロックを解放する。adr は tkse_get_mbk() で得たアドレスでなければならない。
ローカルメモリ空間に確保したメモリ領域を、他のプロセスから解放することはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メモリ状態の参照

tkse_mbk_sts

C言語インタフェース

```
ER ercd = tkse_mbk_sts(M_STATE *pk_sts);
```

パラメータ

M_STATE*	pk_sts	メモリ状態を返す領域

リターンパラメータ

ER	ercd	エラーコード

pk_sts の内容

```
typedef struct  m_state {
    INT    blksz;    /* ブロックサイズ */
    INT    total;    /* 全ブロック数 */
    INT    free;     /* 残りブロック数 */
} M_STATE;
```

blksz	メモリ割り当ての単位(1 ブロック)のバイト数。 実装依存であり、通常は CPU のページサイズとなる。
total	システム全体の総ブロック数。
free	システム全体の未使用ブロック数。

エラーコード

E_OK	正常終了
E_MACV	アドレス(sts)のアクセスは許されていない

解説

現在のメモリ使用状況を取得して pk_sts の示す領域に格納する。
全ブロック数は、現在割り当てられているすべての属性のメモリブロックの合計となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

補足事項

仮想記憶を行っているシステムでは、全ブロック数、残りブロック数を一意に決定できない場合がある。したがって、pk_sts の各要素の具体的な意味は実装に依存する。ただし、free ÷ total が残りメモリの割合の参考値となるような値を返すものとする。

実装により具体的な値を取得できない場合は、全ブロック数、残りブロック数ともに -1 を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.1.3 ライブラリコール

非常駐ローカルメモリの割り当て

malloc

C言語インタフェース

```
void* adr = malloc(size_t size);
```

パラメータ

size_t	size	割り当てる領域のバイト数(> 0)

リターンパラメータ

void*	adr	≠ NULL　正常終了(割り当てたメモリアドレス)
		= NULL　エラー

解説

ローカルメモリ空間に、指定したサイズの非常駐メモリ領域を割り当て、その先頭アドレスを返す。
メモリ領域の割り当てに失敗した場合はNULLを返す。
割り当てたメモリの属性はTA_DELEXITとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

非常駐ローカルメモリの割り当て

calloc

C言語インタフェース

```
void* adr = calloc(size_t nelem, size_t elsize);
```

パラメータ

size_t	nelem	割り当てる要素数(> 0)
size_t	elsize	1要素のバイト数(> 0)

リターンパラメータ

void*	adr	≠ NULL	正常終了(割り当てたメモリアドレス)
		= NULL	エラー

解説

ローカルメモリ空間からelsizeの大きさのnelem個の要素を格納する非常駐メモリ領域を割り当て、その先頭アドレスを返す。

メモリ領域の割り当てに失敗した場合はNULLを返す。

割り当てた領域の内容は0で初期化される。また、割り当てたメモリの属性はTA_DELEXITとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

非常駐ローカルメモリの再割り当て

realloc

C言語インタフェース

```
void*  adr = realloc(void *ptr, size_t size);
```

パラメータ

void	*ptr	サイズを変更する領域のアドレス NULL を指定した場合は新規に割り当てを行う
size_t	size	割り当てるメモリのバイト数(≧0) 0 を指定した場合は領域を解放する

リターンパラメータ

void*	adr	≠ NULL　正常終了(割り当てたメモリアドレス) = NULL　エラー

解説

ローカルメモリ空間に割り当て済みの、ptr で指定した非常駐メモリ領域のサイズを size に変更して再割り当てし、その先頭アドレスを返す。

ptr に NULL を指定した場合は、ローカルメモリ空間に新規にサイズ size の領域を割り当て、その先頭アドレスを返す。

size に 0 を指定した場合は、ptr で指定した領域を解放する。このとき ptr は malloc(),calloc(),realloc()で割り当てたアドレスでなければならない。

ptr = NULL と size = 0 を同時に指定した場合は何も処理せず NULL を返す。

メモリ領域の割り当てに失敗した場合、および領域の解放を指定した場合は NULL を返す。

ptr は NULL、あるいは同じプロセス内で malloc(),calloc(),realloc()で割り当てたアドレスでなければならない。それ以外の値を指定した場合の結果は未定義である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

非常駐ローカルメモリの解放

free

C言語インタフェース

```
void free(void *ptr);
```

パラメータ

void	*ptr	解放する領域のアドレス

リターンパラメータ

なし

解説

ptr で指定したローカルメモリ空間の非常駐メモリ領域を解放する。
ptr に NULL を指定した場合は何も処理しない。
ptr は NULL、あるいは同じプロセス内で malloc(),calloc(),realloc()で割り当てたアドレスでなければならない。それ以外の値を指定した場合の結果は未定義である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

非常駐共有メモリの割り当て

Smalloc

C言語インタフェース

```
void*  adr = Smalloc(size_t size);
```

パラメータ

size_t	size	割り当てる領域のバイト数(> 0)

リターンパラメータ

void*	adr	≠ NULL　正常終了(割り当てたメモリアドレス)
		= NULL　エラー

解説

共有メモリ空間に、指定したサイズの非常駐メモリ領域を割り当て、その先頭アドレスを返す。
メモリ領域の割り当てに失敗した場合はNULLを返す。
割り当てたメモリの属性はTA_DELEXITとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

非常駐共有メモリの割り当て

Scalloc

C言語インタフェース

```
void*  adr = Scalloc(size_t nelem, size_t elsize);
```

パラメータ

size_t	nelem	割り当てる要素数(> 0)
size_t	elsize	1要素のバイト数(> 0)

リターンパラメータ

void*	adr	≠ NULL　正常終了(割り当てたメモリアドレス)
		= NULL　エラー

解説

共有メモリ空間からelsizeの大きさのnelem個の要素を格納する非常駐メモリ領域を割り当て、その先頭アドレスを返す。

メモリ領域の割り当てに失敗した場合はNULLを返す。

割り当てた領域の内容は0で初期化される。また、割り当てたメモリの属性はM_COMMONとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

非常駐共有メモリの再割り当て

Srealloc

C言語インタフェース

```
void*  adr = Srealloc(void *ptr, size_t size);
```

パラメータ

void	*ptr	サイズを変更する領域のアドレス NULL を指定した場合は新規に割り当てを行う
size_t	size	割り当てるメモリのバイト数(≧0) 0 を指定した場合は領域を解放する

リターンパラメータ

void*	adr	≠ NULL	正常終了(割り当てたメモリアドレス)
		= NULL	エラー

解説

共有メモリ空間に割り当て済みの、ptr で指定した非常駐メモリ領域のサイズを size に変更して再割り当てし、その先頭アドレスを返す。

ptr に NULL を指定した場合は、共有メモリ空間に新規にサイズ size の領域を割り当て、その先頭アドレスを返す。

size に 0 を指定した場合は、ptr で指定した領域を解放する。このとき ptr は同じプロセス内で Smalloc(),Scalloc(),Srealloc()で割り当てたアドレスでなければならない。

ptr = NULL と size = 0 を同時に指定した場合は何も処理せず NULL を返す。

メモリ領域の割り当てに失敗した場合、および領域の解放を指定した場合は NULL を返す。

ptr は NULL、あるいは同じプロセス内で Smalloc(),Scalloc(),Srealloc()で割り当てたアドレスでなければならない。それ以外の値を指定した場合の結果は未定義である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

非常駐共有メモリの解放

Sfree

C言語インタフェース

```
void  Sfree(void *ptr);
```

パラメータ

void	*ptr	解放する領域のアドレス

リターンパラメータ

なし

解説

ptr で指定した共有メモリ空間の非常駐メモリ領域を解放する。
ptr に NULL を指定した場合は何も処理しない。
ptr は NULL、あるいは同じプロセス内で Smalloc(),Scalloc(),Srealloc()で割り当てたアドレスでなければならない。それ以外の値を指定した場合の結果は未定義である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.2 プロセス／タスク管理

4.2.1 概要

Standard Extension のプロセス／タスク管理機能は、複数のプロセスが並行動作するシステムを実現するための、各種のプロセス／タスク管理機能を提供する。

プロセス／タスク管理機能は、プロセスやタスクの生成・停止、状態変更、情報取得に関する機能を持つ。プロセス間やタスク間で同期・通信を行う場合は、イベントフラグやメッセージバッファなどのタスク間同期・通信機能や、メッセージ機能、グローバル名、共有メモリなどのプロセス間同期・通信機能を使用する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.2.2 システムコール

プロセスの生成／実行

tkse_cre_prc

C言語インタフェース

```
ER  ercd = tkse_cre_prc(T_CPRC *pk_cprc, MESSAGE* msg);
```

パラメータ

T_CPRC	*pk_cprc	プロセス生成情報
MESSAGE	*msg	プロセス起動時メッセージ

```
typedef struct {
      ATR       prcatr;    /* プロセス属性 */
      VP        prchdr;    /* プロセス生成元オブジェクトへのハンドラ */
      PRI       pri;       /* プロセス優先度 */
                           0 ≦ pri ≦ 255 プロセス優先度を指定値に設定
                           = -1          プロセス優先度を自プロセスと同じ値に設定

        /* その他の実装依存情報 */
} T_CPRC;
```

prcatr はプロセスの属性を表し、以下の指定を行う。

prcatr := (TPA_SYS・TPA_USR) | (TPA_SEIO・TPA_LINK・TPA_PTR)

TPA_SYS	システムプロセスとして生成する
TPA_USR	ユーザプロセスとして生成する
TPA_SEIO	プロセスへのハンドラは、標準入出力のファイルのパス名である
TPA_LINK	プロセスへのハンドラは、標準ファイルシステムのファイルへのリンクである
TPA_PTR	プロセスへのハンドラは、メモリ中に展開されたコードへのポインタである

```
typedef struct {
      W         msg_type;       /* メッセージタイプ */
      W         msg_size;       /* メッセージサイズ(バイト数) */
      MSGBODY   msg_body;       /* メッセージ本体(msg_size バイト) */
} MESSAGE;
```

※MSGBODY 共用体の詳細については、4.2.2 メッセージの構造を参照のこと

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

リターンパラメータ

ER	ercd	≧ 0	正常終了(生成したプロセス ID)
		＜ 0	エラーコード

エラーコード

E_FACV	ファイルのアクセス権(E)がない(TPA_SEIO, TPA_LINK 指定時)
E_MACV	アドレス(msg, hdr(TPA_PTR))のアクセスは許されていない
E_BUSY	ファイルは既に排他的にオープンされているためにオープンできなかった
E_IO	入出力エラーが発生した
E_NOEXS	ファイルは存在していない
E_NOFS	ファイルの属するファイルシステムは接続されていない
E_NOMEM	メモリ領域が不足した(ロードするメモリ領域が不足)
E_REC	ファイルにプログラムレコードが存在しない。またはプログラムレコードの内容が異常である(TPA_LINK 指定時)

解説

プロセスを生成して一意のプロセス ID を割り当てる。

T_CPRC 構造体の prcatr は生成するプロセスの属性を指定する。

TPA_SEIO 属性を指定した場合、指定したファイルの内容をプログラムコードとしてプロセスを生成する。prchdr に対象とするファイルの標準入出力のパス名を文字列で指定する。

TPA_LINK 属性を指定した場合、指定した標準ファイルシステムのファイル内の先頭レコードの内容をプログラムコードとしてプロセス生成する。このとき、先頭レコードのレコードタイプは実行プログラムレコードでなくてはならない。prchdr に対象とする標準ファイルシステムのファイルへのリンク（LINK*）を指定する。

TPA_PTR 属性を指定した場合、メモリ上のプログラムコードを用いてプロセスを生成する。prchdr にメモリ上のプログラムコード領域の先頭アドレスを指定する。なお、メモリ上のプログラムコードの形式は実装依存とする。

プロセスの優先度は pri で指定する。

プロセス生成と同時にメインタスクが起動する。メインタスクの実行開始時に、メインタスク関数の引数に msg で指定したメッセージが渡される。このメッセージの構造は、プロセス間メッセージの構造と同一である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

補足事項

TPA_PTR 属性はプログラムコードの ROM 化を想定している。ROM 上にあるプログラムコードの形式は、ROM 上のデータを直接実行する方法や、ROM 上のデータを一旦 RAM 上に展開してから実行する方法などの複数の方法があり、適用するアプリケーションやハードウェアにより最適な方法は異なる。このため、ROM 上のプログラムコードの形式は実装依存とする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセスの終了

tkse_ext_prc

C言語インタフェース

```
void  tkse_ext_prc(W code);
```

パラメータ

W	code	プロセス終了コード

リターンパラメータ

なし

解説

自プロセスを終了し、指定した code を含むプロセス正常終了メッセージを親プロセスに送信する。

自プロセスで使用中のファイル等のリソースは、一部のもの（生成時に TA_DELEXIT 属性を指定しなかったオブジェクト）を除きすべて自動的に解放される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

他プロセスの強制終了

tkse_ter_prc

C言語インタフェース

```
ER  ercd = tkse_ter_prc(ID pid, W code, W opt);
```

パラメータ

ID	pid	対象プロセス ID ＞ 0　任意のプロセス ＝ -1　親プロセス
W	code	強制終了コード
W	opt	強制終了の指定 (TERM_NRM ・ TERM_ALL) TERM_NRM　指定したプロセスのみ強制終了 TERM_ALL　指定したプロセスおよびそのすべての子孫のプロセスを強制終了

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ILUSE	自プロセスを指定した(pid ＝ 0 または自プロセスの PID)
E_NOEXS	プロセス(pid)は存在していない
E_PAR	パラメータが不正である(opt ＝ TERM_NRM,TERM_ALL 以外を指定した)

解説

指定したプロセスを強制終了し、指定した code を含むプロセス強制終了メッセージを、強制終了したプロセスの親プロセスに送信する。

TERM_ALL を指定した場合は、指定したプロセスおよびそのすべての子孫のプロセスを強制終了する。この場合、指定したプロセスの子孫の強制終了メッセージは送信されない。また、TERM_ALL 指定で自プロセスの親プロセスや、さらに上位の親プロセスを指定した場合は自プロセスも強制終了する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセス／タスクの優先度変更

tkse_chg_pri

C言語インタフェース

```
ER ercd = tkse_chg_pri(ID id, PRI pri, W opt);
```

パラメータ

ID	id	対象プロセス ID またはタスク ID
PRI	pri	変更する優先度
W	opt	優先度変更の指定 (P_ABS ・ P_REL) | [P_TASK] P_ABS　絶対指定（指定した優先度に変更） P_REL　相対指定（現在の優先度 + pri に変更） P_TASK　タスクを対象とする。

リターンパラメータ

ER	ercd	≧ 0　正常終了(変更後の優先度 0～255) P_TASK 指定あり　id のタスクの優先度 P_TASK 指定なし　id のプロセスのメインタスクの優先度 ＜ 0　エラーコード

エラーコード

E_NOEXS	プロセス(id)は存在していない
E_ID	タスク(id)は存在していない。または、自プロセス内のタスクではない
E_PAR	優先度の値が範囲外である(相対指定において新優先度は現在優先度のグループ外) パラメータが不正である(opt ＝ P_ABS,P_REL,P_TASK 以外を指定した)

解説

指定したプロセス／タスクの優先度を変更する。

P_TASK の指定がない場合 :

- id ＝ 0　自プロセス内の全タスクの優先度を変更する。
- id ＝ -1　親プロセス内の全タスクの優先度を変更する。
- id ＞ 0　id で指定したプロセス ID のプロセス内の全タスクの優先度を変更する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

P_TASK を指定した場合 :

id ＝ 0　自タスクの優先度を変更する。

id ＞ 0　id で指定したタスク ID のタスクの優先度を変更する。
指定できるタスクは、自プロセス内のタスクのみである。

P_ABS を指定した場合（絶対指定）は、変更後の優先度は、pri で指定した値となる。

P_REL を指定した場合（相対指定）は、変更後の優先度は、現在の優先度に pri で指定した値が加算された値となる。

相対指定による優先度変更では、現在の優先度グループの範囲外の優先度に変更することはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセス状態の取得

tkse_prc_sts

C言語インタフェース

```
ER ercd = tkse_prc_sts(ID pid, P_STATE* buff, TC* name);
```

パラメータ

ID	pid	対象プロセス ID ＞ 0　任意のプロセス ＝ 0　自プロセス ＝ -1　親プロセス
P_STATE*	buff	プロセス状態の格納領域 （ NULL の場合、格納しない）
TC*	name	プロセス名の格納領域(最大プロセス名+1 文字分の領域) （ NULL の場合、格納しない）

リターンパラメータ

ER	ercd	＞ 0　正常終了(指定プロセス ID) ＜ 0　エラーコード

buff の内容

```
typedef struct {
    UW     state;       /* プロセス状態 */
    PRI    priority;    /* 現在のプロセス優先度( 0 ～ 255 ) */
    ID     parpid;      /* 親プロセスのプロセス ID  */
} P_STATE;
```

name の内容　プロセス名

エラーコード

E_MACV	アドレス(buff,path)のアクセスは許されていない
E_NOEXS	プロセス(pid)は存在していない

解説

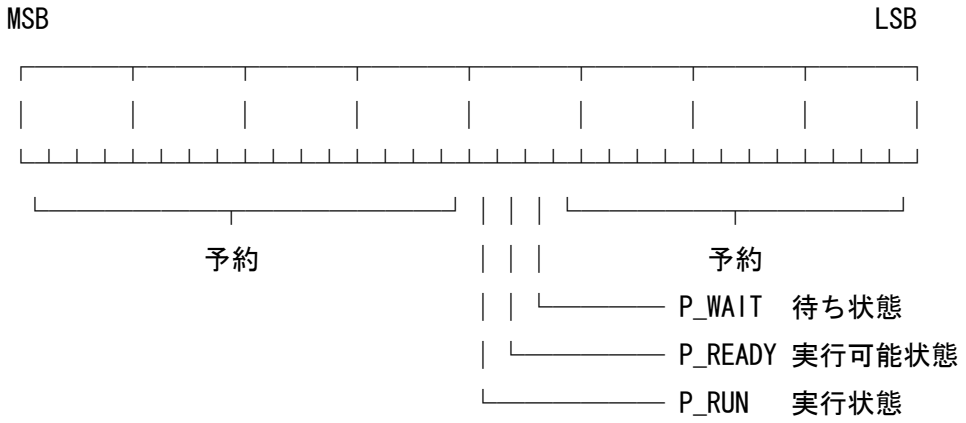
pid で指定したプロセスの状態を取得し、buff で指定した領域に格納する。また指定したプロセスのプロセ

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ス名を name で指定した領域に格納する。buff や name に NULL を指定した場合は、情報の格納は行わない。

プロセス名は、プロセス生成時にシステムが付加する名称である。プロセスが標準ファイルシステムのファイルから生成された場合は、ファイル名がプロセス名となる。その他の場合はシステムが自動的に生成した名前がプロセス名となる。

プロセス状態（ state ）は以下の形式で表す。各状態のいずれかのビットを"1"とすることで、プロセスがその状態にあることを示す。



[図 7] プロセス状態のビット表現

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセス統計情報の取得

tkse_get_inf

C言語インタフェース

```
ER ercd = tkse_get_inf(ID pid, P_INFO* buff);
```

パラメータ

ID	pid	対象プロセス ID ＞ 0　任意のプロセス ＝ 0　自プロセス ＝ -1　親プロセス
P_INFO*	buff	統計情報の格納領域

リターンパラメータ

ER	ercd	エラーコード

buff の内容

```
typedef struct {
    UW      etime;      /* 累計経過時間（秒単位）*/
    UW      utime;      /* プロセスで費やした累計 CPU 時間 */
    UW      stime;      /* システムで費やした累計 CPU 時間 */
    W       tmem;       /* 実行に必要とする全体のメモリサイズ */
    W       wmem;       /* 現在割り付けられている実メモリサイズ */
    W       resv[11];   /* 予約 */
} P_INFO;
```

エラーコード

E_OK	正常終了
E_MACV	アドレス(buff)のアクセスは許されていない
E_NOEXS	プロセス(pid)は存在していない

解説

指定したプロセスの統計情報を取得する。

utime と stime は、システムコール呼び出しの時点でプロセスに存在する、すべてのタスクの実行時間を合計したものとなる。したがって、システムコール呼び出し以前に終了したタスクが費やした時間は含まれない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

utime と stime を合計した値が、そのプロセスが費やした累積 CPU 時間となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセスの終了通知

tkse_req_emg

C言語インタフェース

```
ER ercd = tkse_req_emg(ID pid, W t_mask);
```

パラメータ

ID	pid	対象プロセス ID	
		= -1	親プロセス
		> 0	任意プロセス
W	t_mask	終了通知タイプ指定（以下の値の論理和）	
		0	通知解除
		MM_ABORT	対象プロセスが異常終了したとき通知
		MM_EXIT	対象プロセスが正常終了したとき通知
		MM_TERM	対象プロセスが強制終了したとき通知

リターンパラメータ

ER	ercd	> 0	正常終了(元の t_mask)
		< 0	エラーコード

エラーコード

E_ILUSE	自プロセスを指定した(pid = 0 または自プロセスの PID)
E_NOEXS	プロセス(pid)は存在していない
E_PAR	パラメータが不正である(t_mask が不正)

解説

pid で指定したプロセスの終了時に、終了通知メッセージが自プロセスに送信されるように設定する。
pid= -1 は親プロセスを表す。自プロセスは指定できない(E_ILUSE)。
通知される終了の種類を t_mask で指定する。

```
t_mask = [MM_ABORT] | [MM_EXIT] | [MM_TERM]
```

MM_ABORT	対象プロセスが異常終了したとき通知
MM_EXIT	対象プロセスが正常終了したとき通知
MM_TERM	対象プロセスが強制終了したとき通知

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

t_mask ＝ 0 の時は、自プロセスに設定された終了通知を解除する。t_mask＜0 の時は、現在の設定を変更しない。

戻値に変更前の t_mask の設定値を返す。

終了通知を受け取るよう設定したプロセスが終了した場合は、自動的に設定解除される。

終了通知メッセージは次の形式となる。

```
typedef struct {
        W       type;   /* メッセージタイプ (MS_SYS2) */
        W       size;   /* メッセージサイズ */
        W       kind;   /* 終了タイプ (MS_ABORT,MS_EXIT,MS_TERM) */
        ID      pid;    /* 終了したプロセスのプロセス ID */
        W       code;   /* 終了コード */
} EXITMSG;
```

kind, pid, code は親プロセスに送られる終了メッセージと同じ内容である。

kind　終了メッセージタイプ (MS_ABORT, MS_EXIT, MS_TERM のいずれか)
pid　終了したプロセスのプロセス ID
code　システムのエラーコード、または tkse_ext_prc(), tkse_ter_prc()で指定した終了コード

EXITMSG は MS_SYS2 の各種システムメッセージの中の 1 つとなる。

```
typedef union {
        struct {        /* MS_SYS2 の基本形式 */
                W       type;     /* メッセージタイプ (MS_SYS2) */
                W       size;     /* メッセージサイズ */
                W       kind;     /* 種別 */
                VW      info[1];  /* 種別ごとに異なる各種情報 */
        } base;
        EXITMSG exitmsg;          /* 終了通知 */
} MSG_SYS2;
```

終了通知メッセージは、子プロセスの終了時に親プロセスに通知される終了メッセージと別個に送信される。したがって、あるプロセスが子プロセスに対して終了通知を設定した場合、子プロセスが終了すると、終了通知メッセージ(MS_SYS2)が送られた後、さらに終了メッセージ(MS_ABORT,MS_EXIT,MS_TERM)が送られる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセスの各種情報の取得

tkse_prc_inf

C言語インタフェース

```
ER ercd = tkse_prc_inf(ID pid, W item, VP buf, W len);
```

パラメータ

ID	pid	対象プロセス ID ＝ 0　自プロセス ＝ -1　親プロセス ＞ 0　任意プロセス
W	item	情報の種類 PI_LINK　(0x00010000)　プログラムファイルのリンクの取得 PI_NTSK　(0x00020000)　プロセス内タスク数の取得 PI_TSKSTAT (0x00030000)　各タスクの状態の取得 PI_CREINF　(0x00040000)　プロセス生成時情報
VP	buf	情報格納用バッファ （ NULL の場合、格納しない）
W	len	情報格納用バッファ領域(buf)のバイト長

リターンパラメータ

ER	ercd	＞ 0	正常終了(buf に必要なサイズ(バイト数))
		＜ 0	エラーコード

buf の内容　取得したプロセスの各種情報

エラーコード

E_MACV	アドレス(buff,path)のアクセスは許されていない
E_NOEXS	プロセス(pid)は存在していない
E_PAR	パラメータが不正である(len が不足, item が不正)

解説

pid（プロセス ID）で指定したプロセスに関する各種情報を取得して buf に格納する。
情報の種類は item で指定する。pid ＝ 0 は自プロセス、pid ＝ -1 は親プロセスを示す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

len は buf のサイズ(バイト数)を指定する。必要サイズに満たないときはエラー(E_PAR)となり buf には何も格納されない。

戻値として buf に必要なサイズ(バイト数)を返す。buf ＝ NULL を指定した場合はプロセスに関する情報は格納せず、戻値に buf に必要なサイズのみを返す。この場合 len の指定は無視される。

情報の種類(item)は以下のいずれかを指定する。

```
#define PI_LINK         0x00010000      /* プログラムファイルのリンク */
#define PI_NTSK         0x00020000      /* プロセス内タスク数 */
#define PI_TSKSTAT      0x00030000      /* 各タスクの状態 */
#define PI_CREINF       0x00040000      /* プロセス生成時情報 */
```

PI_LINK :

```
item    PI_LINK
buf     LINK link               プログラムファイルのリンク情報
```

プロセスのプログラムファイルのリンク情報を取得する。
標準ファイルシステムのリンクからプロセスを生成した場合のみ、情報を取得可能である。

PI_NTSK :

```
item    PI_NTSK
buf     W ntsk                  プロセス内のタスク数
```

プロセス内のタスク数(メインタスクとサブタスクの合計)を取得する。

PI_TSKSTAT :

```
item    PI_TSKSTAT + n
buf     P_TSKSTAT tskstat       タスク状態情報

typedef struct {
        ID  tskid;          /* タスク ID */
        UW  state;          /* タスク状態 */
        PRI priority;       /* タスク優先度 */
} P_TSKSTAT;
state :＝ P_DORMANT ・ P_WAIT ・ P_READY ・ P_RUN
```

n 番目のタスクの状態情報を取得する。
n ＝ 0 がメインタスクの情報となり、n ≧ 1 がサブタスクとなる。
n の範囲は PI_NTSK で得られたタスク数 － 1 までが有効となる。

PI_CREINF :

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
item   PI_CREINF
buf    P_CREINF creinf          プロセス生成情報

typedef struc {
       PRI pri;           /* プロセス優先度 */
       ATR prcatr;        /* プロセス属性 */
       VB  prchdr[1];     /* プロセス生成元オブジェクトへのハンドラ */
} P_CREINF;
```

プロセス生成時の情報を取得する。
prchdr[1]のサイズは固定長ではないため、すべての情報を格納するために必要な領域のサイズを取得してから buf を確保し、実際の情報を取得する必要がある。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

サブタスクの生成

tkse_cre_tsk

C言語インタフェース

```
ER ercd = tkse_cre_tsk(FP entry, PRI pri);
```

パラメータ

FP	entry	サブタスク開始アドレス
W	pri	サブタスク優先度

リターンパラメータ

ER	ercd	＞ 0	正常終了(生成したサブタスクのタスク ID)
		＜ 0	エラーコード

エラーコード

E_MACV	アドレス(entry)が不正である
E_LIMIT	サブタスク数の制限を超えた
E_NOMEM	メモリ領域が不足した

解説

自プロセス内のサブタスクを生成する。生成したサブタスクは休止状態となる。
生成に成功すると、生成したサブタスクのタスク ID を返す。
サブタスクは以下の形式の関数とする。

```
void subtask(W arg)
{
    /* プログラムの実行コード */
    tkse_ext_tsk();
}
```

サブタスク関数の引数 arg は、サブタスク起動 tkse_sta_tsk()で指定したサブタスク起動パラメータが渡される。
サブタスクを終了する場合は tkse_ext_tsk()を使用する。サブタスク関数を return で終了してはならない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

サブタスクの起動

tkse_sta_tsk

C言語インタフェース

```
ER ercd = tkse_sta_tsk(ID id, W arg)
```

パラメータ

ID	id	サブタスク ID
W	arg	サブタスク起動パラメータ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	タスク ID(id)が不正あるいは利用できない
E_NOEXS	オブジェクトが存在していない(id のタスクが存在しない)
E_OBJ	オブジェクトの状態が不正（対象タスクが休止状態でない）

解説

tkse_cre_tsk()で生成したサブタスクを起動する。

起動できるのは休止状態のサブタスクのみである。他の状態のタスクを起動しようとした場合はエラーE_OBJを返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

サブタスクの生成・起動

tkse_crs_tsk

C言語インタフェース

```
ER ercd = tkse_crs_tsk(FP entry, PRI pri, W arg);
```

パラメータ

FP	entry	サブタスク開始アドレス
W	pri	サブタスク優先度
W	arg	サブタスク起動パラメータ

リターンパラメータ

ER	ercd	＞ 0	正常終了(生成したサブタスクのタスク ID)
		＜ 0	エラーコード

エラーコード

E_MACV	アドレス(entry)が不正である
E_LIMIT	サブタスク数の制限を超えた
E_NOMEM	メモリ領域が不足した

解説

自プロセス内にサブタスクを生成し起動する。
生成に成功すると、生成したサブタスクのタスク ID を返す。
tkse_cre_tsk()でサブタスクを生成した後に、tkse_sta_tsk()でサブタスクを起動する操作と同等である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

自タスク終了

tkse_ext_tsk

C言語インタフェース

```
void tkse_ext_tsk(void);
```

パラメータ

なし

リターンパラメータ

なし

解説

自タスクを終了する。
メインタスクおよびサブタスクのどちらからでも使用可能である。
メインタスクを終了した場合はプロセスの終了となるため、プロセス内の全タスクが終了する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

他タスク強制終了

tkse_ter_tsk

C言語インタフェース

```
ER ercd = tkse_ter_tsk(ID tskid);
```

パラメータ

ID	tskid	対象タスク ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	タスク ID(tskid)が不正である

解説

指定したタスクを強制終了する。

指定できるタスクは自プロセス内のサブタスクのみである。自タスクまたはメインタスクを指定することはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

タスク起床待ち

tkse_slp_tsk

C言語インタフェース

```
ER ercd = tkse_slp_tsk(TMO tmout);
```

パラメータ

TMO	tmout	タイムアウト時間
		＞ 0　　指定時間（ ミリ秒 ）待つ
		＝ -1　　無限に待つ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_TMOUT	タイムアウト時間が経過したが、起床されなかった
E_DISWAI	メッセージハンドラが起動されたため、待ちが中断された
E_PAR	パラメータ(tmout)が不正である

解説

自タスクを起床待ち状態にする。

tmout で指定したタイムアウト時間が経過する前に、他タスクからこのタスクに tkse_wup_tsk による起床が行われた場合は、待ち状態を解除して正常終了 E_OK を返す。

tmout で指定したタイムアウト時間が経過する間に tkse_wup_tsk による起床が行われなかった場合は、待ち状態を解除してタイムアウトエラーE_TMOUT を返す。

タスクが待ち状態である期間中にメッセージを受信してメッセージハンドラが起動した場合、待ち状態は解除されてエラーE_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

タスク起床

tkse_wup_tsk

C言語インタフェース

```
ER ercd = tkse_wup_tsk(ID tskid);
```

パラメータ

ID	tskid	対象タスク ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	タスク ID(tskid)が不正である
E_LIMIT	起床要求のキューイング数が制限を超えた

解説

tskid で指定したタスクが起床待ち状態である場合、その待ちを解除する。指定したタスクが起床待ち状態でない場合は、起床要求はキューイングされる。

指定できるタスクは自プロセス内のタスクのみである。他プロセスのタスクを起床することはできない。また、自タスクを指定することはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

タスク起床要求のキャンセル

tkse_can_wup

C言語インタフェース

```
ER ercd = tkse_can_wup(ID tskid);
```

パラメータ

ID	tskid	対象タスク ID	
		＞ 0	任意のタスク
		＝ 0	自タスク

リターンパラメータ

ER	ercd	≧ 0	正常終了(キューイングされていた起床要求回数)
		＜ 0	エラーコード

エラーコード

E_ID	タスク ID(tskid)が不正である

解説

tskid で指定したタスクにキューイングされていた起床要求をすべてキャンセルし、キャンセルした起床要求キューイング数を返す。

正常終了すると、キューイングされていた起床要求の回数を返す。

指定できるタスクは、自プロセス内のタスクのみである。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

タスク遅延

tkse_dly_tsk

C言語インタフェース

```
ER ercd = tkse_dly_tsk(RELTIM dlytim);
```

パラメータ

RELTIM	dlytim	遅延時間(ミリ秒≧ 0)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_DISWAI	メッセージハンドラが起動されたため、待ちが中断された
E_PAR	パラメータ(dlytim)が不正である

解説

自タスクを dlytim で指定した遅延時間だけ待ち状態にする。

タスクが待ち状態である期間中にメッセージを受信してメッセージハンドラが起動した場合、待ち状態は解除されてエラーE_DISWAI を返す。

tkse_slp_tsk()とは異なり、dlytim で指定した遅延時間が経過した場合に正常終了 E_OK を返す。また、遅延時間中に tkse_wup_tsk()による起床要求が行われても、待ち解除とはならない。

dlytim = 0 を指定した場合は、自タスクの実行を中断して再スケジューリングを行う。つまり、タスクの状態を実行状態から実行可能状態へ移行させる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

自タスク ID の取得

tkse_get_tid

C 言語インタフェース

```
ER ercd = tkse_get_tid();
```

パラメータ

なし

リターンパラメータ

ER	ercd	＞ 0	正常終了(自タスク ID)
		＜ 0	エラーコード

解説

自タスクのタスク ID を取得する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ロードモジュールのロード

tkse_lod_mod

C言語インタフェース

```
ER ercd = tkse_lod_mod(T_LMOD *pk_mod, P_DYNLDINF *info);
```

パラメータ

T_LMOD *pk_mod	ロードモジュール情報
P_DYNLDINF* info	ロードしたオブジェクトに関する情報

```
typedef struct {
    ATR     modatr;         /* ロードモジュール属性 */
    VP      modhdr;         /* ロードモジュールへのハンドラ */
} T_LMOD;
```

modatr はロードモジュールの属性を表し、以下の指定を行う。

modatr ::= (TMA_SEIO ・ TMA_LINK ・ TMA_PTR)

TMA_SEIO	ロードモジュールへのハンドラは、標準入出力のファイルのパス名である
TMA_LINK	ロードモジュールへのハンドラは、標準ファイルシステムのファイルへのリンクである
TMA_PTR	ロードモジュールへのハンドラは、メモリ中に展開されたコードへのポインタである

```
typedef struct {
    VP      loadaddr;       /* ロードアドレス */
    UW      loadsize;       /* ロードサイズ */
    FP      entry;          /* エントリアドレス */
    UW      info[3];        /* 機種依存の情報 */
} P_DYNLDINF;
```

リターンパラメータ

ER	ercd	＞ 0	正常終了(ロード ID)
		＜ 0	エラーコード

エラーコード

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_FACV	ファイルのアクセス権(E)がない(TMA_SEIO, TMA_LINK 指定時)
E_MACV	アドレス(info, hdr(TMA_PTR))のアクセスは許されていない
E_BUSY	ファイルは既に排他的にオープンされているためにオープンできなかった
E_IO	入出力エラーが発生した
E_NOEXS	ファイルは存在していない
E_NOFS	ファイルの属するファイルシステムは接続されていない
E_NOMEM	メモリ領域が不足した(ロードするメモリ領域が不足)
E_REC	ファイルにプログラムレコードが存在しない。またはプログラムレコードの内容が異常である(TMA_LINK 指定時)

解説

ロードモジュールを自プロセスのローカル空間にロードし、一意のロード ID を割り当てる。

T_LMOD 構造体の modatr はロードモジュールの属性を表す。

TMA_SEIO 属性を指定した場合、指定したファイルの内容をロードモジュールとしてロードする。modhdr に対象とするファイルの標準入出力のパス名を文字列で指定する。

TMA_LINK 属性を指定した場合、指定した標準ファイルシステムのファイル内の最初の実行プログラムレコードの内容をロードモジュールとしてロードする。modhdr に対象とする標準ファイルシステムのファイルへのリンク（LINK*）を指定する。

TMA_PTR 属性を指定した場合、メモリ上のオブジェクトコードをロードモジュールとする。modhdr にメモリ上のオブジェクトコード領域の先頭アドレスを指定する。なお、メモリ上のオブジェクトコードの形式は実装依存とする。

ロードが成功した場合、ロードしたロードモジュールに関する情報を info に返す。

ロードモジュールはメモリ上に単純にロードされる（マッピングされる）だけであり、シンボルアドレスのリロケーション等の処理は行われない。また、すでにロードされているロードモジュールと同じものを指定した場合も、新たに別のメモリ空間を割り当ててロードが行われる。この場合はそれぞれ別個のロード ID が割り当てられる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ロードモジュールのアンロード

tkse_unl_mod

C言語インタフェース

```
ER ercd = tkse_unl_mod(ID loadid);
```

パラメータ

ID	loadid	ロードモジュールのロード ID (tkse_lod_mod() で得られた ID)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	指定したロードモジュールは存在しない

解説

loadid で指定したロードモジュールをアンロードする。ロードモジュールのためにメモリ上にマップされていた領域は、すべてマップ解除される。このときロードモジュールが使用中であるか否かは関知しない。

プロセス終了時には自動的に、そのプロセスがロードしたすべてのロードモジュールがアンロードされる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.3 プロセス間メッセージ

4.3.1 概要

Standard Extension のプロセス間メッセージ機能は、任意のプロセス間においてメッセージの送受信によるプロセス間通信を行うための機能を提供する。

各プロセスには固有のメッセージキューが存在し、このメッセージキューを経由してプロセス間のメッセージ通信が行われる。メッセージ送信先プロセスの指定はプロセス ID を使用する。また、送信元プロセスの識別ももプロセス ID を使用する。

メッセージは、アプリケーションによるプロセス間通信だけではなく、子プロセス終了メッセージなど、システムの情報をアプリケーションのプロセスに通知する用途にも用いられる。

メッセージは他プロセスだけではなく自プロセスにも送信することができる。この種のメッセージとしてタイムアウトメッセージがある。

メッセージキューは FIFO であり、必ず受信順にキューイングされる。メッセージキューがフルの場合は、キューに空きができるまで待つか、エラーリターンするかをメッセージ送信時に指定できる。

受信処理は通常、メッセージキューにキューイングされたメッセージを、メッセージ受信のシステムコールにより取り出す。また、メッセージハンドラを定義することにより、指定したタイプのメッセージを受信した場合に非同期な受信処理を行うことも可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.3.2 メッセージの種類

メッセージはタイプ番号により 1～31 の 31 種のメッセージタイプに分類することができる。
メッセージタイプ番号とそれぞれのメッセージの意味を、以下のように定義する。

```
#define MS_ABORT        ( 1)    /* プロセス異常終了 */
#define MS_EXIT         ( 2)    /* プロセス正常終了 */
#define MS_TERM         ( 3)    /* プロセス強制終了 */
#define MS_TMOUT        ( 4)    /* タイムアウト */
#define MS_SYSEVT       ( 5)    /* システムイベント(強制終了) */
#define MS_SYS1         ( 6)    /* システムで使用 */
#define MS_SYS2         ( 7)    /* システムで使用 */
#define MS_SYS3         ( 8)    /* システムで使用 */
#define MS_SYS4         ( 9)    /* システムで使用 */
#define MS_SYS5         (10)    /* システムで使用 */
#define MS_MNG0         (11)    /* 予約 */
#define MS_MNG1         (12)    /* 予約 */
#define MS_MNG2         (13)    /* 予約 */
#define MS_MNG3         (14)    /* 予約 */
#define MS_MNG4         (15)    /* 予約 */
#define MS_MNG5         (16)    /* 予約 */
#define MS_MNG6         (17)    /* 予約 */
#define MS_MNG7         (18)    /* 予約 */
#define MS_MNG8         (19)    /* 予約 */
#define MS_MNG9         (20)    /* 予約 */
#define MS_MNG10        (21)    /* 予約 */
#define MS_MNG11        (22)    /* 予約 */
#define MS_MNG12        (23)    /* 予約 */
#define MS_TYPE0        (24)    /* アプリケーション定義 */
#define MS_TYPE1        (25)    /* アプリケーション定義 */
#define MS_TYPE2        (26)    /* アプリケーション定義 */
#define MS_TYPE3        (27)    /* アプリケーション定義 */
#define MS_TYPE4        (28)    /* アプリケーション定義 */
#define MS_TYPE5        (29)    /* アプリケーション定義 */
#define MS_TYPE6        (30)    /* アプリケーション定義 */
#define MS_TYPE7        (31)    /* アプリケーション定義 */

#define MS_MIN          ( 1)    /* 最小メッセージタイプ */
#define MS_MAX          (31)    /* 最大メッセージタイプ */
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージタイプ番号0は、プロセス生成時の起動メッセージを受け渡すために、Standard Extensionのシステム内部で使用される。この番号をプロセス間メッセージ機能で直接使用することはできない。

各メッセージタイプにはビット対応のタイプマスクが対応づけられており、タイプマスクの論理和により対象とするメッセージのタイプを複数指定することができる。

メッセージタイプマスクは以下のように定義する。

```
#define MSGMASK(msgtype)        (1 << ((msgtype) - 1))

#define MM_ABORT        MSGMASK(MS_ABORT)       /* プロセス異常終了      */
#define MM_EXIT         MSGMASK(MS_EXIT)        /* プロセス正常終了      */
#define MM_TERM         MSGMASK(MS_TERM)        /* プロセスの強制終了 */
#define MM_TMOUT        MSGMASK(MS_TMOUT)       /* タイムアウト */
#define MM_SYSEVT       MSGMASK(MS_SYSEVT)      /* システムイベント(強制終了) */
#define MM_SYS1         MSGMASK(MS_SYS1)        /* システムで使用 */
#define MM_SYS2         MSGMASK(MS_SYS2)        /* システムで使用 */
#define MM_SYS3         MSGMASK(MS_SYS3)        /* システムで使用 */
#define MM_SYS4         MSGMASK(MS_SYS4)        /* システムで使用 */
#define MM_SYS5         MSGMASK(MS_SYS5)        /* システムで使用 */
#define MM_MNG0         MSGMASK(MS_MNG0)        /* 予約 */
#define MM_MNG1         MSGMASK(MS_MNG1)        /* 予約 */
#define MM_MNG2         MSGMASK(MS_MNG2)        /* 予約 */
#define MM_MNG3         MSGMASK(MS_MNG3)        /* 予約 */
#define MM_MNG4         MSGMASK(MS_MNG4)        /* 予約 */
#define MM_MNG5         MSGMASK(MS_MNG5)        /* 予約 */
#define MM_MNG6         MSGMASK(MS_MNG6)        /* 予約 */
#define MM_MNG7         MSGMASK(MS_MNG7)        /* 予約 */
#define MM_MNG8         MSGMASK(MS_MNG8)        /* 予約 */
#define MM_MNG9         MSGMASK(MS_MNG9)        /* 予約 */
#define MM_MNG10        MSGMASK(MS_MNG10)       /* 予約 */
#define MM_MNG11        MSGMASK(MS_MNG11)       /* 予約 */
#define MM_MNG12        MSGMASK(MS_MNG12)       /* 予約 */

#define MM_TYPE0        MSGMASK(MS_TYPE0)       /* アプリケーション定義 */
#define MM_TYPE1        MSGMASK(MS_TYPE1)       /* アプリケーション定義 */
#define MM_TYPE2        MSGMASK(MS_TYPE2)       /* アプリケーション定義 */
#define MM_TYPE3        MSGMASK(MS_TYPE3)       /* アプリケーション定義 */
#define MM_TYPE4        MSGMASK(MS_TYPE4)       /* アプリケーション定義 */
#define MM_TYPE5        MSGMASK(MS_TYPE5)       /* アプリケーション定義 */
#define MM_TYPE6        MSGMASK(MS_TYPE6)       /* アプリケーション定義 */
#define MM_TYPE7        MSGMASK(MS_TYPE7)       /* アプリケーション定義 */
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
#define MM_ALL          (0x7fffffff)            /* 全マスク */
#define MM_NULL         (0)                     /* 空マスク */
```

4.3.3 メッセージの構造

メッセージは以下の構造を持つ。

```
typedef struct message {
        W       msg_type;       /* メッセージタイプ */
        W       msg_size;       /* メッセージ本体サイズ(バイト) */
        MSGBODY msg_body;       /* メッセージ本体 */
} MESSAGE;
```

メッセージ本体 msg_body の構造は、メッセージタイプ msg_type により異なる。

4.3.4 システムメッセージ

メッセージタイプ番号 1～5 のメッセージをシステムメッセージとする。これは、システムで発生した事象をアプリケーションに通知するためのメッセージである。なお、アプリケーションがシステムメッセージを送信することは特に禁止されていない。

システムメッセージはメッセージキューのオーバーフローの影響を受けず、捨てられることはない。このためシステムメッセージを受信するプロセスは、メッセージの受信を行うことにより、メッセージキューに入れられたシステムメッセージを取り除く必要がある。

それぞれのシステムメッセージの詳細を以下に説明する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

(1) MS_ABORT -- 子プロセスの異常終了メッセージ

プロセスがシステムエラーにより異常終了した場合に、異常終了したプロセスの親プロセスに送信される。

```
W       msg_type :  1  メッセージタイプ (MS_ABORT)
W       msg_size :  8  メッセージ本体のバイト数 (8byte)
MSGBODY msg_body :     メッセージ本体
                    struct {
                            ID      pid;  /* 終了した子プロセスのプロセス ID */
                            W       code;   /* システムエラーコード */
                    };
```

code は発生したシステムエラーのコードであり、MH_TERM メッセージハンドラによる異常終了の場合は、0 となる。

(2) MS_EXIT -- 子プロセスの正常終了メッセージ

プロセスが tkse_ext_prc() システムコールにより正常終了した場合に、正常終了したプロセスの親プロセスに送信される。

```
W       msg_type :  2  メッセージタイプ (MS_EXIT)
W       msg_size :  8  メッセージ本体のバイト数 (8byte)
MSGBODY msg_body :     メッセージ本体
                    struct {
                            ID      pid;  /* 終了した子プロセスのプロセス ID */
                            W       code;   /* tkse_ext_prc()で指定した終了コード */
                    };
```

(3) MS_TERM -- 子プロセスの強制終了メッセージ

プロセスが tkse_ter_prc() システムコールにより他プロセスから強制終了させられた場合に、強制終了したプロセスの親プロセスに送信される。

```
W       msg_type :  3  メッセージタイプ (MS_EXIT)
W       msg_size :  8  メッセージ本体のバイト数 (8byte)
MSGBODY msg_body :     メッセージ本体
                    struct {
                            ID      pid;  /* 終了した子プロセスのプロセス ID */
                            W       code;   /* tkse_ter_prc()で指定した終了コード */
                    };
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

(4) MS_TMOUT -- 自プロセスのタイムアウトメッセージ

tkse_req_tmg() システムコールで要求したタイムアウトメッセージであり、指定時間が経過した後に自プロセスに送信される。

```
W       msg_type :  4  メッセージタイプ (MS_TMOUT)
W       msg_size :  4  メッセージ本体のバイト数 (4byte)
MSGBODY msg_body :     メッセージ本体
                    struct {
                            W       code;   /* tkse_req_tmg()で指定したコード */
                    };
```

4.3.5 メッセージハンドラ

プロセス間メッセージを非同期に受信するための仕組みとして、メッセージハンドラがある。

メッセージハンドラは、登録時に指定したタイプのメッセージをプロセスが受信した際に、現在実行中の処理と非同期に受信処理を行う。メッセージハンドラは登録時にメッセージタイプマスクを指定することにより、メッセージの各タイプに対応した最大 31 種のハンドラを同時に定義可能である。

メッセージハンドラの実行は以下のように行われる。

・登録時に指定したタイプのメッセージを受信した場合、メッセージハンドラはメインタスクの現在の処理に割り込む形で実行される。

・ メインタスクが何らかのシステムコールにより待ち状態である時でも、メッセージを受信した場合はメッセージハンドラが起動する。この場合、メインタスクの待ち状態は無条件で解除されて実行状態となり、待ち状態としていたシステムコールはメッセージハンドラの実行が終了した時点でエラーE_DISWAI を返す。

・ メッセージハンドラは通常のプロセスコードの一部として動作し、実行できるシステムコール等に制限はない。

・ メッセージハンドラの実行が終了すると、通常はメインタスクがハンドラに割り込まれた位置から実行を再開する。また、メッセージハンドラから longjmp()を呼び出すことで、メインタスクの任意の位置（setjmp で指定した位置）に実行を戻すことも可能である。

・ メッセージハンドラはネスティングしない。即ち、新たなメッセージハンドラの起動（別のメッセージタイプのものも含む）は、現在処理中のメッセージハンドラが終了するまで待たされる。

・ メッセージハンドラが起動した場合、その起動の要因となったメッセージはメッセージキューから取り出される。また、取り出されたメッセージのポインタがハンドラのパラメータとして渡される。

・ メッセージハンドラは、必ず tkse_ret_msg()システムコールによって終了しなくてはいけない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージハンドラは、以下の形式の関数として定義する。

```
void msg_hdr(W pid, MESSAGE *r_msg)
{
/*      pid は 送信したプロセスの ID 。(自プロセスの場合は 0) */
/*      r_msg は受信したメッセージへのポインタである。 */

        /* 受信メッセージの処理 */

        tkse_ret_msg (0);  /* 終了 (割り込んだ位置に戻る場合) */

        または

        tkse_ret_msg (1);  /* 終了 (任意の位置に移行する場合) */
        longjmp (reent, code);  /* reent へジャンプする */
}
```

また、メッセージハンドラとして関数アドレスではなく以下の値を指定することもできる。

(1) MH_NONE

受信したメッセージを無視する。メッセージはメッセージキューにキューイングされず、メッセージハンドラも起動しない。この値を指定した場合は例外的に、メッセージ受信時にメインタスクが待ち状態であってもシステムコールは中断されない。

(2) MH_BREAK

受信したメッセージを無視する。メッセージはメッセージキューにキューイングされず、メッセージハンドラも起動しない。ただし MH_NONE とは異なり、メッセージ受信時にメインタスクが待ち状態であった場合はシステムコールは中断しエラーE_DISWAI を返す。この指定は主にタイムアウト等の処理に使用される。

(3) MH_TERM

自プロセスを異常終了し、MS_ABORT メッセージ（エラーコードは 0）を親プロセスに送信する。

初期プロセスは MH_NONE 以外のメッセージハンドラを使用することはできない。MH_NONE 以外を指定した場合はメッセージを受信してもハンドラは実行されない。また MH_BREAK および MH_TERM の指定も無視される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.3.6 システムコール

メッセージの送信

tkse_snd_msg

C言語インタフェース

```
ER ercd = tkse_snd_msg(ID pid, MESSAGE* msg, W opt);
```

パラメータ

ID	pid	送信先プロセス ID ＞ 0　任意のプロセス ＝ 0　自プロセス ＝-1　親プロセス
MESSAGE*	msg	送信メッセージ
W	opt	送信待ちの指定 (NOWAIT ・ WAIT ・ CONFM) NOWAIT : メッセージキューの空きを待たない WAIT　: メッセージキューの空きを待つ CONFM　: メッセージの受信を待つ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(msg)のアクセスは許されていない
E_DISWAI	メッセージハンドラが起動されたため待ち処理が中断された
E_NOEXS	プロセス(pid)は存在していない
E_PAR	パラメータが不正である(オプション、メッセージタイプが不正)
E_ILUSE	自プロセスを指定した(pid ＝ 0 または自プロセスの PID)(CONFM 指定時)
E_LIMIT	メッセージ本体のサイズがシステムの制限を越えた、または 0 以下である
E_SYSMEM	システムのメモリ領域が不足した(送信先のメッセージキューがフル(NOWAIT 指定時))。

解説

pid で指定したプロセスにメッセージ msg を送信する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

opt はメッセージ送信時の送信待ちの挙動を指定する。

opt = NOWAIT を指定した場合は、メッセージを送信先プロセスのメッセージキューに入れた時点で正常終了する。送信先のメッセージキューに空きがない時はエラー終了する。

opt = WAIT を指定した場合は、メッセージを送信先プロセスのメッセージキューに入れた時点で正常終了する。メッセージキューに空きがない時は空きができるまで待つ。受信処理を待っている間に送信先プロセスが終了した場合はエラーを返す。

opt = CONFM を指定した場合は、メッセージを送信先プロセスのメッセージキューに入れ、送信先プロセスが送信したメッセージを受信、またはメッセージキューからクリアされるまで待ち、メッセージが処理された時点で正常終了する。ただし tkse_rcv_msg() で CHECK 指定によりメッセージヘッダ部分のみを得た場合には、受信したとみなされず待ちは解除されない。また、NOCLR 指定により受信した場合は、送信したメッセージがメッセージキューに残っていてもメッセージが受信されたとみなし、正常終了する。受信処理を待っている間に送信先プロセスが終了した場合はエラー終了する。送信先が自プロセスの場合、CONFM 指定はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージの受信

tkse_rcv_msg

C言語インタフェース

```
ER ercd = tkse_rcv_msg(W t_mask, MESSAGE* msg, W msgsz, W opt);
```

パラメータ

W	t_mask	受信対象メッセージタイプマスク
MESSAGE*	msg	受信メッセージ格納領域
W	msgsz	受信メッセージ格納領域全体のバイトサイズ。メッセージのヘッダ部分も含むため msgsz ≧ 8 でなくてはいけない。
W	opt	受信動作の指定 (WAIT ・ NOWAIT ・ WAIEVT)	(CLR ・ NOCLR)	(CHECK) WAIT　: 指定タイプのメッセージの受信待ち NOWAIT : 指定タイプのメッセージの受信待ち無し WAIEVT : 指定タイプのメッセージの受信、およびイベント発生待ち CLR　: メッセージの受信後、そのメッセージをキューから取り除く NOCLR　: メッセージの受信後、そのメッセージをキューに残す CHECK :　メッセージの有無をチェック

リターンパラメータ

ER	ercd	＞ 0　正常終了(受信メッセージの送信元プロセス ID) ＝ 0　正常終了(受信メッセージの送信元は自プロセス) ＜ 0　エラーコード

エラーコード

E_MACV	アドレス(msg)のアクセスは許されていない
E_DISWAI	メッセージハンドラが起動されたため待ち処理が中断された
E_TMOUT	指定したタイプ(t_mask)のメッセージは存在しない(NOWAIT 指定時)
E_PAR	パラメータが不正である(msgsz が小さすぎる、WAIEVT 指定時以外 t_mask≦0、WAIEVT 指定時 t_mask<0)

解説

自プロセスに送信されたメッセージのうち、指定したタイプ t_mask のものを受信し、msg で指定した領域に格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

受信したメッセージがメッセージサイズ msgsz より大きい場合は、msgsz で指定したサイズのみを msg の領域に格納してエラーを返す。この場合は CLR 指定があってもメッセージはメッセージキューに残る。ただし、msgsz < 8 の場合は msg の領域には何も格納せずエラーを返す。メッセージ全体を取得できなかった場合は、取得したメッセージのヘッダ部分から実際のメッセージサイズを取り出し、メッセージが十分に格納できる領域を確保してから再度 tkse_rcv_msg()を実行する。

opt はメッセージ受信時の挙動を指定する。

opt ＝ WAIT を指定した場合は、指定タイプのメッセージを受信するまで待つ。

opt ＝ NOWAIT を指定した場合は、指定タイプのメッセージがメッセージキューに存在した場合は正常終了し、存在していない場合は直ちにエラー終了する。

opt ＝ WAIEVT を指定した場合は、指定タイプのメッセージを受信するまで待つ。ただし WAIT 指定とは異なり、メッセージを受信待ちの間に tkse_brk_msg() によりイベント発生が通知されると待ちは解除され、エラー終了する。

WAIEVT を指定した場合のみ、t_mask ＝ 0 を指定可能である。この場合、メッセージの受信は行われず、イベント発生が通知されるまで待ち状態となる。

WAIEVT を指定できるのはシステム全体で同時に 1 つのタスクのみである。複数のタスクが WAIEVT 指定で tkse_rcv_msg()を呼び出した場合は、最後に呼び出したタスクの WAIEVT 指定のみが有効となり、他のタスクは WAIT と同じ扱いとなる。

WAIEVT は Standard Extension より上位のシステムで使用することを前提としている。通常はアプリケーションから WAIEVT 指定を使用してはならない。

opt ＝ CLR を指定した場合は、メッセージ受信後に該当するメッセージをキューから取り除く。

opt ＝ NOCLR を指定した場合は、メッセージ受信後も該当するメッセージをキューに残す。

opt ＝ CHECK を指定した場合、以下のような動作になる。

1. メッセージキューにメッセージが存在しない場合、WAIT または WAIEVT 指定時は待ち状態となる。NOWAIT 指定時はエラー終了する。

2. メッセージキューに指定したタイプのメッセージが存在する場合は、msg にメッセージを格納し正常終了する。CLR 指定であればキューから取り除き、NOCLR 指定であればキューに残す。

3. メッセージキューに指定したタイプのメッセージが存在せず、それ以外のタイプのメッセージが存在する場合は、メッセージのヘッダ領域である先頭 8 バイト（msg_type と msg_size）のみを msg に格納し正常終了する。この場合、CLR／NOCLR の指定に関わらずメッセージはキューに残す。

CHECK を指定した場合は、指定したタイプ以外のメッセージを取得する可能性があるため、受信したメッセージのタイプを必ずチェックする必要がある。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージのクリア

tkse_clr_msg

C言語インタフェース

```
ER ercd = tkse_clr_msg(W t_mask, W last_mask);
```

パラメータ

W	t_mask	クリア対象メッセージタイプマスク (MM_ALL を指定した場合は、全メッセージタイプ)
W	last_mask	クリア終了メッセージタイプマスク (MM_NULL を指定した場合は対象はメッセージキューの最後まで) (MM_ALL を指定した場合は、メッセージを１つだけクリア)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である(t_mask≦0、last_mask<0)

解説

自プロセス宛の指定したタイプの受信済みメッセージをクリアする。

自プロセスのメッセージキューに存在するメッセージのうち、t_mask で指定したタイプのメッセージを last_mask で指定したタイプのメッセージの直前までクリアする。last_mask で指定したタイプのメッセージはクリアされないが、last_mask ＝ MM_ALL の時は、メッセージを１つだけクリアする。

t_mask と last_mask の指定例を以下に示す。

t_mask	last_mask	動作
MM_ALL	MM_NULL	受信済みの全メッセージをクリア
－	MM_ALL	t_mask で指定したメッセージを１つだけクリア
MM_ALL	MM_ALL	先頭のメッセージを１つだけクリア

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

タイムアウトメッセージの要求

tkse_req_tmg

C言語インタフェース

```
ER ercd = tkse_req_tmg(TMO tmout, W code);
```

パラメータ

TMO	tmout	メッセージ送信時間(ミリ秒)
W	code	タイムアウトメッセージのコード

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である(time≦0)
E_SYSMEM	システムのメモリ領域が不足した

解説

tmout で指定した時間後に、自プロセスにタイムアウトメッセージ（MS_TMOUT）を送信するよう要求する。
この機能は、メッセージハンドラと組み合わせて特定の処理のタイムアウトを監視する場合などに使用する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

タイムアウトメッセージの取り消し

tkse_can_tmg

C言語インタフェース

```
ER ercd = tkse_can_tmg();
```

パラメータ

なし

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了

解説

自プロセスのタイムアウトメッセージの要求をすべて取り消す。タイムアウトメッセージの要求がない場合は何もせず正常終了する。

すでに送信されてメッセージキューに入っているタイムアウトメッセージはクリアされない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベント発生の通知

tkse_brk_msg

C言語インタフェース

```
ER ercd = tkse_brk_msg();
```

パラメータ

なし

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了

解説

WAIEVT属性指定によるtkse_rcv_msg()の待ちを解除する。

tkse_brk_msg() を呼び出した際にWAIEVT指定で待ち状態となっているタスクが存在しなければ、待ち解除の要求を記録する。ただし、待ち解除の要求回数は記録しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージハンドラの定義

tkse_def_msg

C言語インタフェース

```
ER ercd = tkse_def_msg(W t_mask, FUNCP msg_hdr);
```

パラメータ

W	t_mask	対象メッセージタイプマスク
FUNCP	msg_hdr	メッセージハンドラ開始アドレス
	NULL	メッセージハンドラの定義解除
	MH_NONE	システム定義ハンドラ(無視)
	MH_BREAK	システム定義ハンドラ(処理中断)
	MH_TERM	システム定義ハンドラ(プロセス終了)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(msg_hdr)のアクセスは許されていない
E_PAR	パラメータが不正である(t_mask≦0)

解説

t_mask で指定したメッセージタイプのメッセージに対応するメッセージハンドラ msg_hdr を定義する。

同じメッセージタイプのメッセージハンドラがすでに定義されている場合には、後から定義したメッセージハンドラが有効となる。

msg_hdr ＝ NULL の場合は定義されたメッセージハンドラを解除する。

初期プロセスは MH_NONE 以外のメッセージハンドラを使用することはできない。MH_NONE 以外を指定した場合はメッセージを受信してもハンドラは実行されない。また MH_BREAK および MH_TERM の指定も無視される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージハンドラの終了

tkse_ret_msg

C言語インタフェース

```
ER ercd = tkse_ret_msg(W ret)
```

パラメータ

W	ret	リターン指定
	=0	メッセージハンドラが割り込んだ位置から実行を再開する
	≠0	システムコールから戻り、そのまま実行を続行する

リターンパラメータ

ER	ercd	エラーコード
		(ret= 0 の時)リターンしない
		(ret≠ 0 の時)=0　正常

エラーコード

E_OK	正常終了

解説

メッセージハンドラの実行を終了する。

ret = 0 を指定した場合、メインタスクの実行をメッセージハンドラが割り込んだ位置から再開する。この場合 tkse_ret_msg()からは戻らない。待ちを含むシステムコールの実行中に割り込んだ場合、メッセージハンドラが起動されたことを示すエラーコードがそのシステムコールから戻る。システムコールの実行は保証されない。

ret ≠ 0 を指定した場合、メインタスクの実行はメッセージハンドラが割り込んだ位置からは再開せず、tkse_ret_msg()から戻ってそのまま実行を続行する。この場合、通常はハンドラの最後で longjmp() により制御を他の場所へ移す。

ret の指定に関わらず、メッセージハンドラの処理の最後で必ず実行しなくてはならない。また、メッセージハンドラ以外の場所で実行してはならない。実行した場合の動作は未定義である。

メッセージハンドラの起動要求が多重に発生していた場合、tkse_ret_msg()の実行後に、要求のあったメッセージハンドラが起動する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.4 グローバル名

4.4.1 概要

Standard Extension のグローバル名機能は、プロセス間で共有するデータに任意の名前を付与する機能、および付与した名前からデータを参照する機能を提供する。

グローバル名で共有するデータは 1 つの 32 ビットデータ(W 型)である。データに付与する名前は TNULL(0)で終了する TRON コード文字列で、最大 256 文字（512 バイト）までである。名前は通常、TRON コードとして意味のあるものを使用するが、終端が TNULL であれば TRON コード以外のデータを名前に使用しても構わない。
グローバル名で共有するデータはすべてのプロセスから参照・変更可能である。また、グローバル名を生成したプロセス以外のプロセスからの変更・削除を禁止することもできる。

グローバル名の主な用途としては以下のデータの共有が挙げられるが、これら以外の用途に用いることも可能である。

・プロセス ID
・共有メモリ領域のアドレス
・セマフォやメッセージバッファ、ランデブなどの同期・通信オブジェクト ID
・システム全体で使用される環境パラメータ

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.4.2 システムコール

グローバル名データ生成

tkse_cre_nam

C言語インタフェース

```
ER ercd = tkse_cre_nam(TC* name, W data, W opt);
```

パラメータ

TC*	name	対象グローバル名(先頭256文字(512バイト)までが有効)
W	data	登録するデータ
W	opt	データ生成の指定
	(N_CREATE ・ N_MODIFY ・ N_FORCE) | [NA_PROTECT] | [TA_DELEXIT]	
	N_CREATE : 新規生成	
	N_MODIFY : 変更	
	N_FORCE : 生成および変更	
	NA_PROTECT : 変更・削除の保護指定	
	TA_DELEXIT : 自動削除の指定	

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(name)のアクセスは許されていない
E_OBJ	グローバル名(name)は既に存在している(N_CREATEの時)
	グローバル名(name)は保護されている(N_MODIFY, N_FORCEの時)
E_NOEXS	グローバル名(name)は存在していない(N_MODIFYの時)
E_PAR	パラメータが不正である(optが不正,nameが空)
E_SYSMEM	システムのメモリ領域が不足した

解説

name で指定した名前のグローバル名データを生成、または変更する。

opt はグローバル名の生成・変更の挙動を指定する。

opt ＝ N_CREATE を指定した場合は、指定した名前のグローバル名データが存在していない場合に生成する。すでに存在している場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

opt ＝ N_MODIFY を指定した場合は、指定した名前のグローバル名データが存在している場合にそのデータを変更する。存在していない場合はエラーとなる。
opt ＝ N_FORCE を指定した場合は、指定した名前のグローバル名データが存在していない場合に生成する。すでに存在している場合はそのデータを変更する。

N_CREATE, N_FORCE と共に NA_PROTECT を指定することができる。NA_PROTECT を指定した場合、そのグローバル名データは、生成したプロセス以外からの変更・削除を禁止される。NA_PROTECT が指定されたグローバル名データに対し、生成したプロセス以外から tkse_cre_nam()または tkse_del_nam()が発行された場合、E_OBJ エラーとなる。NA_PROTECT の指定はグローバル名データが削除されるまで有効である。

他のオプションと共に TA_DELEXIT を指定することができる。TA_DELEXIT を指定した場合は、生成したプロセス、または最後にデータを変更したプロセスの終了時にグローバル名データを自動削除する。既に他のプロセスから TA_DELEXIT 指定が行われていた場合でも、最後に TA_DELEXIT 属性でグローバル名を生成・変更したプロセスが自動削除の対象となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

グローバル名データ削除

tkse_del_nam

C言語インタフェース

```
ER ercd = tkse_del_nam(TC* name);
```

パラメータ

TC*	name	対象グローバル名(先頭256文字(512バイト)までが有効)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(name)のアクセスは許されていない
E_OBJ	グローバル名(name)は保護されている
E_NOEXS	グローバル名(name)は存在していない
E_PAR	パラメータが不正である(nameが空)

解説

nameで指定したグローバル名データを削除する。

グローバル名データ生成時にNA_PROTECTを指定していた場合、そのグローバル名データを生成したプロセス以外から削除しようとするとE_OBJエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

グローバル名データ取得

tkse_get_nam

C言語インタフェース

```
ER ercd = tkse_get_nam(TC* name, W* data);
```

パラメータ

TC*	name	対象グローバル名(先頭256文字(512バイト)までが有効)
W*	data	取得データ格納領域

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(name,data)のアクセスは許されていない
E_NOEXS	グローバル名(name)は存在していない
E_PAR	パラメータが不正である(nameが空)

解説

nameで指定したグローバル名データを取得する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.5 タスク間同期・通信

4.5.1 概要

Standard Extension のタスク間同期・通信機能は、複数のタスク間で同期や通信を実現するための機能を提供する。提供する機能として以下のものがある。

・セマフォ
・ミューテックス
・イベントフラグ
・メールボックス
・メッセージバッファ
・ランデブポート

それぞれの機能は、T-Kernel の対応する機能とほぼ同等である。ただし、機能の一部に制限があるため完全な互換ではない。

セマフォなどの各オブジェクトの ID は Standard Extension で管理する ID であるため、T-Kernel のシステムコールで使用する ID とは異なる。このため Standard Extension で作成したオブジェクトの ID を T-Kernel でそのまま使用することはできない。また、T-Kernel で作成したオブジェクトを Standard Extension で扱うこともできない。ただし、例外としてタスク ID は Standard Extension と T-Kernel で同一となる。

以降の説明では T-Kernel システムコールとの違い（制限事項）について記述する。各システムコールの詳細は T-Kernel 仕様書を参照のこと。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.5.2 システムコール（セマフォ）

セマフォの生成

tkse_cre_sem

C言語インタフェース

```
ID semid = tkse_cre_sem( T_CSEM *pk_csem );
```

パラメータ

T_CSEM	*pk_csem	セマフォ生成情報

```
typedef struct t_csem {
        VP      exinf;          /* 拡張情報 */
        ATR     sematr;         /* セマフォ属性 */
        INT     isemcnt;        /* セマフォの初期カウント値 */
        INT     maxsem;         /* セマフォの最大カウント値 */
} T_CSEM;
```

セマフォ属性 sematr

sematr:= (TA_TFIFO ・ TA_TPRI) | (TA_FIRST ・ TA_CNT) | TA_DELEXIT

TA_ TFIFO	待ちタスクを FIFO で管理
TA_TPRI	待ちタスクを優先度順で管理
TA_FIRST	待ち行列先頭のタスクを優先
TA_CNT	要求数の少ないタスクを優先
TA_DELEXIT	自動削除の指定

リターンパラメータ

ID	semid	＞ 0	セマフォ ID（正常終了）
		＜ 0	エラーコード

エラーコード

E_NOMEM	メモリ不足(管理ブロック用の領域が確保できない)
E_LIMIT	セマフォの数がシステムの上限を超えた
E_RSATR	予約属性(sematr が不正あるいは利用できない)
E_PAR	パラメータエラー(pk_csem が不正、isemcnt, maxsem が負または不正)

解説

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

pk_csem に従いセマフォを生成する。ただし exinf の指定は無視される。

sematr は生成するセマフォの属性を指定する。TA_DELEXIT 属性を指定した場合、セマフォを生成したプロセスが終了すると自動的にセマフォが削除される。

その他の属性は T-Kernel のセマフォと同等である。ただし、TA_NODISWAI 属性の指定はできない。

生成したセマフォは、すべてのプロセスから使用可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

セマフォの削除

tkse_del_sem

C言語インタフェース

```
ER ercd = tkse_del_sem( ID semid );
```

パラメータ

ID	semid	セマフォ ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(semid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(semid のセマフォが存在しない)

解説

semid で指定したセマフォを削除する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

セマフォの資源返却

tkse_sig_sem

C言語インタフェース

```
ER ercd = tkse_sig_sem ( ID semid, INT cnt );
```

パラメータ

ID	semid	セマフォ ID
INT	cnt	返却資源数

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(semid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(semid のセマフォが存在しない)
E_QOVR	キューイングまたはネストのオーバーフロー(キューイング数 semcnt のオーバーフロー)
E_PAR	パラメータエラー(cnt≦0)

解説

semid で指定したセマフォに cnt 個の資源を返却する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

セマフォの資源獲得

tkse_wai_sem

C言語インタフェース

```
ER ercd = tkse_wai_sem( ID semid, INT cnt, TMO tmout );
```

パラメータ

ID	semid	セマフォ ID
INT	cnt	返却資源数
TMO	tmout	タイムアウト時間

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(semid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(semid のセマフォが存在しない)
E_PAR	パラメータエラー(tmout≦(-2), cnt≦0)
E_DLT	待ちオブジェクトが削除された(待ちの間に対象セマフォが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト

解説

semid で指定したセマフォから cnt 個の資源を獲得する。
セマフォ獲得待ち状態の間にメッセージハンドラ処理が割り込んだ場合、待ち状態は解除され E_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

セマフォの状態参照

tkse_ref_sem

C言語インタフェース

```
ER ercd = tkse_ref_sem ( ID semid, T_RSEM *pk_rsem );
```

パラメータ

T_RSEM	*pk_rsem	セマフォの状態を返すアドレス

```
typedef struct t_rsem {
        VP      exinf;          /* 拡張情報 */
        ID      wtsk;           /* 待ちタスクの ID */
        INT     semcnt;         /* 現在のセマフォカウント値 */
} T_RSEM;
```

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(semid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(semid のセマフォが存在しない)
E_PAR	パラメータエラー(リターンパラメータ用のパケットアドレスが使用できない値)

解説

semid で指定したセマフォの状態を参照し、pk_rsem で指定したアドレスにその内容を返す。
ただし exinf は生成時の指定に関わらず常に NULL となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.5.3 システムコール（ミューテックス）

ミューテックスの生成

tkse_cre_mtx

C言語インタフェース

```
ID mtxid = tkse_cre_mtx( T_CMTX *pk_cmtx );
```

パラメータ

T_CMTX *pk_cmtx　ミューテックス生成情報

```
typedef struct t_cmtx {
        VP      exinf;          /* 拡張情報 */
        ATR     mtxatr;         /* ミューテックス属性 */
        PRI     ceilpri;        /* ミューテックスの上限優先度 */
} T_CMTX;
```

ミューテックス属性 mtxatr

mtxatr :＝ (TA_TFIFO ・ TA_TPRI) | TA_DELEXIT

TA_TFIFO	待ちタスクを FIFO で管理
TA_TPRI	待ちタスクを優先度順で管理
TA_DELEXIT	自動削除の指定

リターンパラメータ

ID	mtxid	＞ 0	ミューテックス ID（正常終了）
		＜ 0	エラーコード

エラーコード

E_NOMEM	メモリ不足(管理ブロック用の領域が確保できない)
E_LIMIT	ミューテックスの数がシステムの上限を超えた
E_RSATR	予約属性(mtxatr が不正あるいは利用できない)
E_PAR	パラメータエラー(pk_cmtx,ceilpri が不正)

解説

pk_cmtx に従いミューテックスを生成する。ただし exinf, ceilpri の指定は無視される。

mtxatr はミューテックスの属性を指定する。TA_DELEXIT 属性を指定した場合、ミューテックスを生成した

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

プロセスが終了すると自動的にミューテックスが削除される。

その他の属性は T-Kernel のミューテックスと同等である。ただし、TA_NODISWAI 属性、TA_INHERIT 属性、TA_CEILING 属性の指定はできない。

生成したミューテックスは、すべてのプロセスから使用可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ミューテックスの削除

tkse_del_mtx

C言語インタフェース

```
ER ercd = tkse_del_mtx ( ID mtxid )
```

パラメータ

ID	mtxid	ミューテックス ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mtxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mtxid のミューテックスが存在しない)

解説

mtxid で指定したミューテックスを削除する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ミューテックスのロック

tkse_loc_mtx

C言語インタフェース

```
ER ercd = tkse_loc_mtx( ID mtxid, TMO tmout );
```

パラメータ

ID	mtxid	ミューテックス ID
TMO	tmout	タイムアウト時間

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mtxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mtxid のミューテックスが存在しない)
E_PAR	パラメータエラー(tmout≦(-2))
E_DLT	待ちオブジェクトが削除された(待ちの間に対象ミューテックスが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト
E_ILUSE	不正使用(多重ロック、上限優先度違反)

解説

mtxid で指定したミューテックスをロックする。

ミューテックスのロック待ち状態の間にメッセージハンドラ処理が割り込んだ場合、待ち状態は解除され E_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ミューテックスのアンロック

tkse_unl_mtx

C言語インタフェース

```
ER ercd = tkse_unl_mtx( ID mtxid );
```

パラメータ

ID	mtxid	ミューテックス ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mtxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mtxid のミューテックスが存在しない)
E_ILUSE	不正使用(自タスクがロックしたミューテックスではない)

解説

mtxid で指定したミューテックスをアンロックする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ミューテックスの状態参照

tkse_ref_mtx

C言語インタフェース

```
ER ercd = tkse_ref_mtx( ID mtxid, T_RMTX *pk_rmtx );
```

パラメータ

ID	mtxid	ミューテックス ID
T_RMTX	*pk_rmtx	ミューテックスの状態を返すアドレス

```
typedef struct t_rmtx {
        VP      exinf;          /* 拡張情報 */
        ID      htsk;           /* ロックしているタスクの ID */
        ID      wtsk;           /* ロック待ちタスクの ID */
} T_RMTX;
```

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mtxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mtxid のミューテックスが存在しない)
E_PAR	パラメータエラー(リターンパラメータ用のパケットアドレスが使用できない値)

解説

mtxid で指定したミューテックスの状態を参照し、pk_rmtx で指定したアドレスにその内容を返す。
ただし exinf は生成時の指定に関わらず常に NULL となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.5.4 システムコール（イベントフラグ）

イベントフラグの生成

tkse_cre_flg

C言語インタフェース

```
ID flgid = tkse_cre_flg( T_CFLG *pk_cflg )
```

パラメータ

T_CFLG *pk_cflg　　イベントフラグ生成情報

```
typedef struct t_cflg {
        VP      exinf;          /* 拡張情報 */
        ATR     flgatr;         /* イベントフラグ属性 */
        UINT    iflgptn;        /* イベントフラグの初期値 */
} T_CFLG;
```

イベントフラグ属性 flgatr

flgatr:= (TA_TFIFO ・ TA_TPRI) | (TA_WMUL ・ TA_WSGL) | TA_DELEXIT

TA_TFIFO	待ちタスクを FIFO で管理
TA_TPRI	待ちタスクを優先度順で管理
TA_WSGL	複数タスクの待ちを許さない
TA_WMUL	複数タスクの待ちを許す
TA_DELEXIT	自動削除の指定

リターンパラメータ

ID	flgid	＞ 0	イベントフラグ ID（正常終了）
		＜ 0	エラーコード

エラーコード

E_NOMEM	メモリ不足(管理ブロック用の領域が確保できない)
E_LIMIT	イベントフラグの数がシステムの上限を超えた
E_RSATR	予約属性(flgatr が不正あるいは利用できない)
E_PAR	パラメータエラー(pk_cflg が不正)

解説

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

pk_cflg に従いイベントフラグを生成する。ただし exinf の指定は無視される。

flgatr は生成するイベントフラグの属性を指定する。TA_DELEXIT 属性を指定した場合、イベントフラグを生成したプロセスが終了すると自動的にイベントフラグが削除される。

その他の属性は T-Kernel のイベントフラグと同等である。ただし、TA_NODISWAI 属性の指定はできない。

生成したイベントフラグは、すべてのプロセスから使用可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントフラグの削除

tkse_del_flg

C言語インタフェース

```
ER ercd = tkse_del_flg ( ID flgid );
```

パラメータ

ID	flgid	イベントフラグ ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(flgid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(flgid のイベントフラグが存在しない)

解説

flgid で指定したイベントフラグを削除する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントフラグのセット

tkse_set_flg

C言語インタフェース

```
ER ercd = tkse_set_flg( ID flgid, UINT setptn );
```

パラメータ

ID	flgid	イベントフラグ ID
UINT	setptn	セットするビットパターン

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(flgid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(flgid のイベントフラグが存在しない)

解説

flgid で指定したイベントフラグに setptn のパターンをセットする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントフラグのクリア

tkse_clr_flg

C言語インタフェース

```
ER ercd = tkse_clr_flg( ID flgid, UINT clrptn );
```

パラメータ

ID	flgid	イベントフラグ ID
UINT	clrptn	クリアするビットパターン

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(flgid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(flgid のイベントフラグが存在しない)

解説

flgid で指定したイベントフラグを clrptn のパターンでクリアする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントフラグ待ち

tkse_wai_flg

C言語インタフェース

```
ER ercd = tkse_wai_flg( ID flgid, UINT waiptn, UINT wfmode, UINT *p_flgptn, TMO tmout );
```

パラメータ

ID	flgid	イベントフラグ ID
UINT	waiptn	待ちビットパターン
UINT	wfmode	待ちモード
	TWF_ANDW	AND 待ち
	TWF_ORW	OR 待ち
	TWF_CLR	全クリア指定
	TWF_BITCLR	条件ビットのみクリア指定
UINT	*p_flgptn	待ち解除時のビットパターンを返すアドレス
TMO	tmout	タイムアウト時間

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(flgid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(flgid のイベントフラグが存在しない)
E_PAR	パラメータエラー(waiptn＝0,wfmode が不正,tmout≦(-2))
E_OBJ	オブジェクトの状態が不正(TA_WSGL 属性のイベントフラグに対する複数タスクの待ち)
E_DLT	待ちオブジェクトが削除された(待ちの間に対象イベントフラグが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト

解説

flgid で指定したイベントフラグに､wfmode で指定した待ち条件で waiptn で指定したビットがセットされ

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

るのを待つ。

　イベントフラグ待ち状態の間にメッセージハンドラ処理が割り込んだ場合、待ち状態は解除され E_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントフラグの状態参照

tkse_ref_flg

C言語インタフェース

```
ER ercd = tkse_ref_flg( ID flgid, T_RFLG *pk_rflg );
```

パラメータ

T_RFLG *pk_rflg		イベントフラグの状態を返すアドレス

```
typedef struct t_rflg {
        VP      exinf;          /* 拡張情報 */
        ID      wtsk;           /* 待ちタスクの ID */
        UINT    flgptn;         /* 現在のイベントフラグパターン */
} T_RFLG;
```

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(flgid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(flgid のイベントフラグが存在しない)
E_PAR	パラメータエラー(リターンパラメータ用のパケットアドレスが使用できない値)

解説

flgid で指定したイベントフラグの状態を参照し、pk_rflg で指定したアドレスにその内容を返す。
ただし exinf は生成時の指定に関わらず常に NULL となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.5.5 システムコール（メールボックス）

メールボックス生成

tkse_cre_mbx

C言語インタフェース

```
ID mbxid = tkse_cre_mbx( T_CMBX *pk_cmbx );
```

パラメータ

T_CMBX* pk_cmbx　メールボックス生成情報

```
typedef struct t_cmbx {
    VP      exinf;      /* 拡張情報 */
    ATR     mbxatr;     /* メールボックス属性 */
} T_CMBX;
```

メールボックス属性 mbxatr

mbxatr:= (TA_TFIFO ・ TA_TPRI) | (TA_MFIFO ・ TA_MPRI)

TA_TFIFO	待ちタスクのキューイングは FIFO
TA_TPRI	待ちタスクのキューイングは優先度順
TA_MFIFO	メッセージのキューイングは FIFO
TA_MPRI	メッセージのキューイングは優先度順

リターンパラメータ

ID	mbxid	＞ 0	メールボックス ID（正常終了）
		＜ 0	エラーコード

エラーコード

E_NOMEM	メモリ不足(管理ブロックやバッファ用の領域が確保できない)
E_LIMIT	メールボックスの数がシステムの上限を超えた
E_RSATR	予約属性(mbxatr が不正あるいは利用できない)
E_PAR	パラメータエラー(pk_cmbx が不正)

解説

pk_cmbx に従いメールボックスを生成する。ただし exinf の指定は無視される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

mbxatr は生成するメールボックスの属性を指定する。属性の指定は T-Kernel のメールボックスと同等である。ただし TA_NODISWAI 属性の指定はできない。また、 TA_DELEXIT 属性を指定することはできず、指定した場合は無視される。

メールボックスはデータをポインタで渡すため、メールボックスを生成したプロセス内のタスク間でのみ使用可能である。所属するプロセスの異なるタスク間の通信に使用することはできない。そのため、メールボックスを生成したプロセスが終了した場合、メールボックスは自動的に削除される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メールボックス削除

tkse_del_mbx

C言語インタフェース

```
ER ercd = tkse_del_mbx( ID mbxid ) ;
```

パラメータ

ID	mbxid	メールボックス ID	

リターンパラメータ

ER	ercd	≧ 0	正常終了
		< 0	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mbxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbxid のメールボックスが存在しない)

解説

mbxid で指定したメールボックスを削除する。
メールボックスを生成したプロセス以外から、メールボックスを削除しようとした場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メールボックスへ送信

tkse_snd_mbx

C言語インタフェース

```
ER  ercd = tkse_snd_mbx( ID mbxid, T_MSG *pk_msg );
```

パラメータ

ID	mbxid	メールボックス ID
T_MSG*	pk_msg	メッセージパケットの先頭アドレス

リターンパラメータ

ER	ercd	≧ 0	正常終了
		< 0	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mbxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbxid のメールボックスが存在しない)
E_PAR	パラメータエラー(pk_msg が使用できない値)

解説

mbxid で指定したメールボックスに、pk_msg を先頭アドレスとするメッセージパケットを送信する。
メッセージパケットの内容はコピーされず、受信時には先頭アドレス(pk_msg の値)のみが渡される。

メールボックスを生成したプロセス以外から、メッセージを送信した場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メールボックスから受信

tkse_rcv_mbx

C言語インタフェース

```
ER ercd = tkse_rcv_mbx( ID mbxid, T_MSG **ppk_msg, TMO tmout ) ;
```

パラメータ

ID	mbxid	メールボックス ID
TMO	tmout	タイムアウト指定

リターンパラメータ

ER	ercd	≧ 0	正常終了
		＜ 0	エラーコード
T_MSG*	pk_msg	メッセージパケットの先頭アドレス	

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mbxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbxid のメールボックスが存在しない)
E_PAR	パラメータエラー(tmout≦(-2))
E_DLT	待ちオブジェクトが削除された(待ちの間に対象メールボックスが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト

解説

mbxid で指定したメールボックスからメッセージを受信する。
メールボックスを生成したプロセス以外から、メッセージを受信しようとした場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メールボックスの状態参照

tkse_ref_mbx

C言語インタフェース

```
ER tkse_ref_mbx( ID mbxid, T_RMBX *pk_rmbx ) ;
```

パラメータ

ID mbxid	メールボックス ID	
T_RMBX* pk_rmbx	メールボックス状態を返すパケットアドレス	

```
typedef struct t_rmbx {
        VP exinf;       /* 拡張情報 */
        ID wtsk;        /* 待ちタスクの有無 */
        T_MSG* pk_msg;  /* 次に受信されるメッセージパケットの先頭アドレス */
} T_RMBX;
```

リターンパラメータ

ER	ercd	≧ 0	正常終了
		＜ 0	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mbxid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbxid のメールボックスが存在しない)
E_PAR	パラメータエラー(リターンパラメータ用のパケットアドレスが使用できない値)

解説

semid で指定したメールボックスの状態を参照し、pk_rmbx で指定したアドレスにその内容を返す。
ただし exinf は生成時の指定に関わらず常に NULL となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.5.6 システムコール（メッセージバッファ）

メッセージバッファの生成

tkse_cre_mbf

C言語インタフェース

```
ID mbfid = tkse_cre_mbf( T_CMBF *pk_cmbf );
```

パラメータ

```
T_CMBF *pk_cmbf        メッセージバッファ生成情報
    typedef struct t_cmbf {
            VP      exinf;          /* 拡張情報 */
            ATR     mbfatr;         /* メッセージバッファ属性 */
            INT     bufsz;          /* メッセージバッファのサイズ(バイト) */
            INT     maxmsz;         /* メッセージの最大長(バイト) */
    } T_CMBF;

    メッセージバッファ属性 mbfatr
        mbfatr := (TA_TFIFO | TA_TPRI) | TA_DELEXIT
            TA_TFIFO        待ちタスクを FIFO で管理
            TA_TPRI         待ちタスクを優先度順で管理
            TA_DELEXIT      自動削除の指定
```

リターンパラメータ

```
ID          mbfid       > 0     メッセージバッファ ID（正常終了）
                        < 0     エラーコード
```

エラーコード

```
E_NOMEM     メモリ不足(管理ブロックやリングバッファ用の領域が確保できない)
E_LIMIT     メッセージバッファの数がシステムの上限を超えた
E_RSATR     予約属性(mbfatr が不正あるいは利用できない)
E_PAR       パラメータエラー(pk_cmbf が不正,bufsz,maxmsz が負または不正)
```

解説

pk_cmbf に従いメッセージバッファを生成する。ただし exinf の指定は無視される。
mbfatr はメッセージバッファの属性を指定する。TA_DELEXIT 属性を指定した場合、メッセージバッファを

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

生成したプロセスが終了すると自動的にメッセージバッファが削除される。

その他の属性は T-Kernel のメッセージバッファと同等である。ただし、TA_NODISWAI 属性の指定はできない。

生成したメッセージバッファは、すべてのプロセスから使用可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージバッファの削除

tkse_del_mbf

C言語インタフェース

```
ER ercd = tkse_del_mbf( ID mbfid );
```

パラメータ

ID	mbfid	メッセージバッファ ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mbfid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbfid のメッセージバッファが存在しない)

解説

mbfid で指定したメッセージバッファを削除する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージバッファへ送信

tkse_snd_mbf

C言語インタフェース

```
ER ercd = tkse_snd_mbf( ID mbfid, VP msg, INT msgsz, TMO tmout );
```

パラメータ

ID	mbfid	メッセージバッファ
VP	msg	送信メッセージの先頭アドレス
INT	msgsz	送信メッセージのサイズ（バイト数）
TMO	tmout	タイムアウト時間

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mbfid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbfid のメッセージバッファが存在しない)
E_PAR	パラメータエラー(msgsz≦0,msgsz＞maxmsz,msg が使用できない値,tmout≦(-2))
E_DLT	待ちオブジェクトが削除された(待ちの間に対象メッセージバッファが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト

解説

mbfid で指定したメッセージバッファに、msg と msgsz で指定したメッセージを送信する。

メッセージ送信待ち状態の間にメッセージハンドラ処理が割り込んだ場合、待ち状態は解除され E_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージバッファから受信

tkse_rcv_mbf

C言語インタフェース

```
INT msgsz = tkse_rcv_mbf( ID mbfid, VP msg, TMO tmout );
```

パラメータ

ID	mbfid	メッセージバッファ
VP	msg	受信メッセージを格納する先頭アドレス
TMO	tmout	タイムアウト時間

リターンパラメータ

INT	msgsz	＞ 0	受信したメッセージのサイズ（バイト数）
		＜ 0	エラーコード

エラーコード

E_ID	不正 ID 番号(mbfid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbfid のメッセージバッファが存在しない)
E_PAR	パラメータエラー(msg が使用できない値,tmout≦(-2))
E_DLT	待ちオブジェクトが削除された(待ちの間に対象メッセージバッファが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト

解説

mbfid で指定したメッセージバッファからメッセージを受信して msg に格納する。

メッセージ受信待ち状態の間にメッセージハンドラ処理が割り込んだ場合、待ち状態は解除され E_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

メッセージバッファの状態参照

tkse_ref_mbf

C言語インタフェース

```
ER ercd = tkse_ref_mbf( ID mbfid, T_RMBF *pk_rmbf );
```

パラメータ

ID	mbfid	メッセージバッファ
T_RMBF *pk_rmbf		メッセージバッファ状態情報

```
typedef struct t_rmbf {
        VP      exinf;          /* 拡張情報 */
        ID      wtsk;           /* 受信待ちタスクの ID */
        ID      stsk;           /* 送信待ちタスクの ID */
        INT     msgsz;          /* 次に受信されるメッセージのサイズ(バイト) */
        INT     frbufsz;        /* 空きバッファのサイズ(バイト) */
        INT     maxmsz;         /* メッセージの最大長(バイト) */
} T_RMBF;
```

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(mbfid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(mbfid のメッセージバッファが存在しない)
E_PAR	パラメータエラー(リターンパラメータ用のパケットアドレスが使用できない値)

解説

mbfid で指定したメッセージバッファの状態を参照し、pk_rmbf で指定したアドレスにその内容を返す。
ただし exinf は生成時の指定に関わらず常に NULL となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.5.7 システムコール（ランデブポート）

ランデブポートの生成

tkse_cre_por

C言語インタフェース

```
ID porid = tkse_cre_por( T_CPOR *pk_cpor );
```

パラメータ

T_CPOR *pk_cpor　ランデブポートの生成情報

```
typedef struct t_cpor {
        VP      exinf;          /* 拡張情報 */
        ATR     poratr;         /* ポート属性 */
        INT     maxcmsz;        /* 呼出メッセージの最大長(バイト) */
        INT     maxrmsz;        /* 返答メッセージの最大長(バイト) */
} T_CPOR;
```

ランデブポート属性 poratr

poratr:= (TA_TFIFO ・ TA_TPRI) | TA_DELEXIT

TA_TFIFO	待ちタスクを FIFO で管理
TA_TPRI	待ちタスクを優先度順で管理
TA_DELEXIT	自動削除の指定

リターンパラメータ

ID	porid	＞ 0	ランデブポート ID（正常終了）
		＜ 0	エラーコード

エラーコード

E_NOMEM	メモリ不足(管理ブロック用の領域が確保できない)
E_LIMIT	ランデブポートの数がシステムの上限を超えた
E_RSATR	予約属性(poratr が不正あるいは利用できない)
E_PAR	パラメータエラー(pk_cpor が不正,maxcmsz,maxrmsz が負または不正)

解説

pk_cpor に従いランデブポートを生成する。ただし exinf の指定は無視される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

poratr はランデブポートの属性を指定する。TA_DELEXIT 属性を指定した場合、ランデブポートを生成したプロセスが終了すると自動的にランデブポートが削除される。

その他の属性は T-Kernel のランデブポートと同等である。ただし TA_NODISWAI 属性の指定はできない。

生成したランデブポートは、すべてのプロセスから使用可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ランデブポートの削除

tkse_del_por

C言語インタフェース

```
ER ercd = tkse_del_por( ID porid );
```

パラメータ

ID	porid	ランデブポート ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(porid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(porid のランデブポートが存在しない)

解説

porid で指定したランデブポートを削除する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ランデブの呼び出し

tkse_cal_por

C言語インタフェース

```
INT rmsgsz = tkse_cal_por( ID porid, UINT calptn, VP msg, INT cmsgsz, TMO tmout );
```

パラメータ

ID	porid	ランデブポート ID
UINT	calptn	選択条件を表すビットパターン
VP	msg	メッセージの先頭アドレス
INT	cmsgsz	呼び出しメッセージのサイズ（バイト数）
TMO	tmout	タイムアウト時間

リターンパラメータ

INT	rmsgsz	＞ 0	返答メッセージのサイズ（バイト数）
		＜ 0	エラーコード

エラーコード

E_ID	不正 ID 番号(porid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(porid のランデブポートが存在しない)
E_PAR	パラメータエラー(cmsgsz＜0,cmsgsz＞maxcmsz,calptn＝0,msg が使用できない値, tmout≦(-2))
E_DLT	待ちオブジェクトが削除された(待ちの間に対象ランデブポートが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト

解説

porid で指定したランデブポートに対するランデブ呼び出しを行う。
ランデブ呼出待ち状態の間にメッセージハンドラ処理が割り込んだ場合、待ち状態は解除され E_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ランデブの受け付け

tkse_acp_por

C言語インタフェース

```
INT cmsgsz = tkse_acp_por( ID porid, UINT acpptn, RNO *p_rdvno, VP msg, TMO tmout );
```

パラメータ

ID	porid	ランデブポート ID
UINT	acpptn	選択条件を表すビットパターン
INT	*p_rdvno	ランデブ番号を返すアドレス
VP	msg	メッセージの先頭アドレス
TMO	tmout	タイムアウト時間

リターンパラメータ

INT	cmsgsz	＞ 0	呼び出しメッセージのサイズ（バイト数）
		＜ 0	エラーコード

エラーコード

E_ID	不正 ID 番号(porid が不正あるいは利用できない,porid が他ノードのランデブポート)
E_NOEXS	オブジェクトが存在していない(porid のランデブポートが存在しない)
E_PAR	パラメータエラー(acpptn＝0,msg が使用できない値,tmout≦(-2))
E_DLT	待ちオブジェクトが削除された(待ちの間に対象ランデブポートが削除)
E_RLWAI	待ち状態強制解除
E_DISWAI	待ち禁止による待ち解除
E_TMOUT	ポーリング失敗またはタイムアウト

解説

porid で指定したランデブポートに対するランデブ受け付けを行う。
ランデブ受付待ち状態の間にメッセージハンドラ処理が割り込んだ場合、待ち状態は解除され E_DISWAI を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ランデブの回送

tkse_fwd_por

C言語インタフェース

```
ER ercd = tkse_fwd_por( ID porid, UINT calptn, RNO rdvno, VP msg, INT cmsgsz );
```

パラメータ

ID	porid	ランデブポート ID
UINT	calptn	選択条件を表すビットパターン
INT	rdvno	回送前のランデブ番号
VP	msg	メッセージの先頭アドレス
INT	cmsgsz	呼び出しメッセージのサイズ（バイト数）

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(porid が不正あるいは利用できない,porid が他ノードのランデブポート)
E_NOEXS	オブジェクトが存在していない(porid のランデブポートが存在しない)
E_PAR	パラメータエラー(cmsgsz＜0,cmsgsz＞回送後の maxcmsz,cmsgsz＞回送前の maxrmsz, calptn＝0,msg が使用できない値)
E_OBJ	オブジェクトの状態が不正(rdvno が不正,回送後の maxrmsz＞回送前の maxrmsz)
E_DISWAI	待ち禁止による待ち解除

解説

一旦受け付けたランデブを他のランデブポートに回送する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ランデブの返答

tkse_rpl_rdv

C言語インタフェース

```
ER ercd = tkse_rpl_rdv( RNO rdvno, VP msg, INT rmsgsz );
```

パラメータ

INT	rdvno	ランデブ番号
VP	msg	返答メッセージの先頭アドレス
INT	rmsgsz	返答メッセージのサイズ（バイト数）

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータエラー(rmsgsz＜0,rmsgsz＞maxrmsz,msg が使用できない値)
E_OBJ	オブジェクトの状態が不正(rdvno が不正)

解説

ランデブ相手のタスクに返答を返し、ランデブを終了する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ランデブポートの状態参照

tkse_ref_por

C言語インタフェース

```
ER ercd = tkse_ref_por( ID porid, T_RPOR *pk_rpor );
```

パラメータ

ID	porid	ランデブポート ID
T_RPOR *pk_rpor		ランデブポート状態情報

```
typedef struct t_rpor {
        VP      exinf;          /* 拡張情報 */
        ID      wtsk;           /* 呼出待ちタスクの ID */
        ID      atsk;           /* 受付待ちタスクの ID */
        INT     maxcmsz;        /* 呼出メッセージの最大長(バイト) */
        INT     maxrmsz;        /* 返答メッセージの最大長(バイト) */
} T_RPOR;
```

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	不正 ID 番号(porid が不正あるいは利用できない)
E_NOEXS	オブジェクトが存在していない(porid のランデブポートが存在しない)
E_PAR	パラメータエラー(リターンパラメータ用のパケットアドレスが使用できない値)

解説

porid で指定したランデブポートの状態を参照し、pk_rpor で指定したアドレスにその内容を返す。
ただし exinf は生成時の指定に関わらず常に NULL となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6 標準入出力

4.6.1 概要

Standard Extension の標準入出力機能(Standard Extension Input Output)は、主としてファイル操作に関連する入出力機能を提供する。標準入出力機能は標準ファイル管理機能とは異なり、さまざまな仕様のファイルシステムを統一的に扱うために、以下のシステムコールを用いてファイル操作を行う。

ER tkse_attach(const TC *devnm, const char *connm, int mode)	ファイルシステムの接続
ER tkse_detach(const TC *devnm, int eject)	ファイルシステムの切断
ER tkse_open(const char *path, int oflag, mode_t mode)	ファイルシステムのオープン
ER tkse_close(int fildes)	ファイルシステムのクローズ
ER tkse_lseek(int fildes, off_t offset, int whence)	ファイル／ディレクトリの現在位置の移動
ER tkse_read(int fildes, void *buf, size_t nbyte)	ファイルの読み込み
ER tkse_write(int fildes, const void *buf, size_t nbyte)	ファイルの書き込み
ER tkse_getdents(int fildes, struct dirent *buf, size_t nbyte)	ディレクトリエントリの取り出し
ER tkse_stat(const char *path, struct stat *sb)	ファイル情報の取得 1
ER tkse_lstat(const char *path, struct stat *sb)	ファイル情報の取得 2
ER tkse_fstat(inf fildes, struct stat *sb)	ファイル情報の取得 3
ER tkse_rename(const char *from, const char *to)	ファイル名の変更
ER tkse_unlink(const char *path)	ディレクトリエントリの削除
ER tkse_mkdir(const char *path, mode_t mode)	ディレクトリの作成
ER tkse_rmdir(const char *path)	ディレクトリの削除
ER tkse_dup(int oldd)	ファイルディスクリプタの複製 1
ER tkse_dup2(int oldd, int newd)	ファイルディスクリプタの複製 2
ER tkse_fsync(int fildes)	ファイルのディスクキャッシュ内容とディスクの同期
ER tkse_chdir(const char *path)	カレントディレクトリの変更 1
ER tkse_fchdir(int fildes)	カレントディレクトリの変更 2
ER tkse_chmod(const char *path, mode_t mode)	ファイルモードの変更 1
ER tkse_fchmod(int fildes, mode_t mode)	ファイルモードの変更 2
ER tkse_creat(const char *path, mode_t mode)	ファイルの作成
ER tkse_utimes(const char *path, const struct timeval times[2])	アクセス時間、修正時間の変更
ER tkse_umask(mode_t cmask)	ファイル作成マスクの設定
ER tkse_truncate(const char *path, off_t length)	ファイルサイズを指定長に設定 1
ER tkse_ftruncate(int fildes, off_t length)	ファイルサイズを指定長に設定 2
ER tkse_sync(void)	ディスクキャッシュの内容とディスクの同期
ER tkse_getfsstat(struct statfs *buf, W bufsize, int flags)	ファイルシステムのリストの取得
ER tkse_getlink(const char *path, char *buf)	標準ファイルの LINK の取得

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6.2 対応ファイルシステム

標準入出力で扱うことができるファイルシステムは、以下の３種類である。

(1) T-Kernel 標準ファイルシステム

T-Kernel 標準ファイルシステムはファイルとディレクトリの区別が存在しない。このため標準入出力上では次の条件により、標準ファイルシステム上のファイルをディレクトリ／ファイルとして識別する。

・ディレクトリ
　リンクレコードを含むファイル
　ファイルタイプ(ファイルのアプリケーションタイプ)が 6 のファイル

・ファイル
　ディレクトリ以外のファイル

※ファイルの操作対象は、レコードタイプ 31 のレコードの内の先頭のレコード１つのみとなる。操作対象レコードは、tkse_open または、ファイル作成後初めての tkse_write の時点で確定し、tkse_close するまで変わることはない。

(2) FAT ファイルシステム

FAT12, FAT16, FAT32 のファイルシステムに対応する。また、VFAT のロングファイル名も使用可能である。
フロッピーディスクなどの区画情報を持たないディスク、およびハードディスクなどの区画情報を持つディスクの両方にアクセス可能である。ただし、区画情報は基本区画のみサポートとする。

(3) CD-ROM ファイルシステム

ISO 9660 Level 1 のファイルシステムに対応する。ただし読み込みのみで書き込みはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6.3 ファイル操作

(1) ファイルシステム接続／切断

標準入出力でデバイス上のファイルシステムにアクセスために、まずファイルシステムの接続(tkse_attach)を行う。このとき指定するファイルシステムの名称を接続名と呼ぶ。
接続を解除する場合は、ファイルシステムの切断(tkse_detach)を行う。

(2) ファイルのオープン／クローズ

ファイルシステム接続後､ファイルのオープン(tkse_open)によりファイルの書き込み･読み込みが可能となる。ファイルのオープンが成功すると、オープン操作に対応するファイルディスクリプタが新規に割り当てられる。ファイルディスクリプタは 0 以上の整数値を持つ識別子であり、この識別子を用いてファイル操作を行う。
ファイルのクローズ(tkse_close)により、ファイルディスクリプタは無効となる。
ファイルディスクリプタは該当ファイルをオープンしたプロセス内でのみ有効である。他プロセスがオープンしたファイルディスクリプタを用いてファイル操作を行うことはできない。
プロセスの終了により、そのプロセスがオープンしていたファイルはすべてクローズされる。
ディレクトリもファイルと同様、オープン／クローズでファイルディスクリプタを得ることができる。

4.6.4 ファイルディスクリプタの初期状態

プロセスを起動すると、以下のファイルディスクリプタをオープンした状態となる。

STDIN_FILENO	0	標準入力
STDOUT_FILENO	1	標準出力
STDERR_FILENO	2	標準エラー出力

これらのファイルディスクリプタは、起動したプロセスに割り当てられるコンソール I/O となる。
通常のファイルディスクリプタと異なりクローズすることはできない。

4.6.5 ディスクキャッシュ

ファイルの書き込み・読み込みを効率的に行うため、ディスクキャッシュを使用する。ファイルのクローズ、またはディスクキャッシュの同期(tkse_sync, tkse_fsync)を行うことで、ディスクキャッシュのデータをデバイス上のファイルに反映させることができる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6.6 ファイル名

ファイル名の文字コードは日本語 EUC（ASCII コードを含む）を使用する。日本語 EUC で表すことのできない文字をファイル名に指定することはできない。ファイルシステム上に日本語 EUC で表すことのできない名称を持つファイルが存在した場合、その名称を取得した結果は未定義である。

ファイル名の最大長はファイルシステムにより異なる。いずれのファイルシステムにおいても、その最大長を越えた部分の名称は無視される。

・ファイルを参照する場合
　最大長を越えた分を無視してファイル名の一致を検索する。

・ファイルを生成する場合
　最大長を越えた分を切り捨てたファイル名で生成する。

・ファイル名の取得
　最大長を越えた分を切り捨てたファイル名を得る。

※T-Kernel 標準ファイルシステムを使用する場合にのみ、以下の独自仕様を適用する。

(1) 文字コード変換

T-Kernel 標準ファイルシステムはファイル名として TRON コードを使用するため、標準入出力の内部で日本語 EUC と TRON コードの変換を行う。
日本語 EUC から TRON コードに変換する場合、文字列に半角文字が含まれる場合は、対応する全角文字(JIS 第１水準)に変換される。
TRON コードから日本語 EUC に変換する場合、文字列に半角文字で表記可能な TRON コード文字が含まれる場合は、対応する半角文字に変換される。ただし、"／" および ":" の２つの文字はパス名および出現順の区切りとして用いるため、半角に変換されない。

例えば「例題１」というファイルの場合、ファイル名を取得すると"１"が半角文字"1"に変換されて"例題1" という文字列を得る。また、このファイルを指定する場合、"例題 1"あるいは"例題１"のどちらのファイル名を使用しても「例題１」を指定することができる。
「原稿用紙／１版」というファイルの場合、"/"はパス名の区切りであるため変換対象とはならない。このファイルのファイル名を取得すると"原稿用紙／1 版"となる。

(2) ファイル名最大長の拡張

T-Kernel 標準ファイルシステムのファイル名は最大 20 文字であるが、ファイル名のエンコードの方式を特殊

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

なものとすることで、ASCII 文字のみで構成されるファイル名を最大 34 文字に拡張することができる。

ファイル名が 20 文字を越え、かつそのファイル名がすべて ASCII 文字(ただし、0x20～0x7e の範囲の文字のみ)の場合に以下の特殊エンコード方式を適用する。それ以外の場合は、単純に日本語 EUC から TRON コードへ変換してファイル名とする。

・特殊エンコード方式

T-Kernel 標準ファイルシステムにおけるファイル名は 1 文字が 2 バイト(TC 型 ＝ TRON コード)で構成される。この 2 バイト／文字で構成されるファイル名の先頭 3 文字を特殊エンコード方式の開始マークとし、残りの 17 文字分 34 バイトを 1 バイト／文字のファイル名として構成する。

・特殊エンコード方式の開始マーク

TK_U(0x2355), TK_X(0x2358), 0xA121

・ファイル名のエンコード

ASCII コード(0x20～0x7e)を 0x80～0xde に変換し 2 文字ずつパックして、TC 型(2 バイト／文字の形式)に変換する。

((c1 + 0x60) << 8) | (c2 + 0x60) → TC 型

c1: 奇数文字目
c2: 偶数文字目

ファイル名の文字数が奇数の場合、最後の 1 文字はパックせずに通常の TRON コード(TC 型)とする。

※0xA121 は JIS X 0212(補助漢字)の 1 区 1 点に相当する。JIS では未定義の文字である。
※パック後は TRON コードのＤゾーン内の未定義文字と KSC5601(韓国) に相当する文字になる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6.7 パス名

パス名はファイルの場所を示す文字列である。パス名の文字列は、ルートディレクトリを起点として、該当するファイルに到達するまでのディレクトリツリーの経路を順番に並べたものである。

/接続名/ディレクトリ名/ファイル名
(例) "/SYS/DIR/FILE.EXT"

文字コードは日本語 EUC を使用する。区切りコード "/"、"." は、半角文字(ASCII)とする。
T-Kernel 標準ファイルシステムでは ":1" のような出現順の指定も可能である。

(例) "/SYS/DIR:1/FILE. EXT:2"
":" および数字は半角文字(ASCII)とする。

※ パス名中に出現した ":" 以降のすべての文字が数字の時、その数字が示す数値が出現順とみなされる。また、出現順は既に存在するファイルを検索する場合にのみ適用され、新規ファイル作成時は出現順は無視される。
例えば、NEW_FILE:3 というファイル名を指定した場合、検索時は出現順 3 の NEW_FILE が対象となり、新規ファイル作成時は NEW_FILE というファイル名のファイルが作成される。

※ パス名指定時はカレントディレクトリの機能をサポートしない。常にフルパス名で指定する必要がある。

4.6.8 ルートディレクトリ

ルートディレクトリは、他のすべてのディレクトリの上位に位置する仮想的なディレクトリである。ルートディレクトリの実体は、ファイルシステム上には存在しない。
ルートディレクトリの直下には、現在接続されている全ファイルシステムが仮想的なサブディレクトリとして存在する。各サブディレクトリの名称はファイルシステムの接続名となる。ファイルシステムを接続するとルートディレクトリの下に新たな仮想サブディレクトリが作成され、この仮想サブディレクトリの下に実際のファイルやディレクトリが配置される。
ルートディレクトリは読み込み専用のディレクトリであり、通常のファイルやディレクトリをルートディレクトリの直下に作成することはできない。また、ファイル名変更のシステムコールを使用して、ルートディレクトリ直下の仮想サブディレクトリの名称（接続名）を変更することはできない。

ルートディレクトリは、ディレクトリのオープンとクローズ、ディレクトリエントリの取り出し、ディレクトリ情報の取得を行うことができる。また、カレントディレクトリをルートディレクトリに移動することも可能である。その他のシステムコールでルートディレクトリを指定した場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6.9 カレントディレクトリ

プロセスはそれぞれ個別にカレントディレクトリ情報を保持する。カレントディレクトリは相対パス名によるファイル／ディレクトリアクセスを実現するために用いられる。

プロセスのカレントディレクトリ情報は子プロセスに引き継がれる。また、初期プロセスのカレントディレクトリは、初期状態でルートディレクトリ ″/″ となる。

いずれかのプロセスがカレントディレクトリに設定しているディレクトリは削除することはできない。ただしディレクトリ名の変更は可能である。

4.6.10 自ディレクトリ ″.″ と親ディレクトリ ″..″

パス名として自ディレクトリ ″.″ と親ディレクトリ ″..″を使用することができる。

自ディレクトリはカレントディレクトリを示し、親ディレクトリはカレントディレクトリのひとつ上のディレクトリを示す。ただし例外として、ルートディレクトリの親ディレクトリはルートディレクトリ自身を示す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6.11 エラーコード

標準入出力ライブラリを使用した場合、エラー発生時に以下のエラーコードを変数 errno に設定する。

```
#include <errno.h>
```

EFAULT	不正アドレス
EINVAL	不正パラメータ
ENOMEM	メモリ不足
EEXIST	すでに存在している
EMFILE	オープン可能なファイル数を越えた
ESRCH	プロセスがない
EINTR	システムコールが割り込まれた
EBADF	不正ファイルディスクリプタ
EACCES	アクセス権がない
EPERM	許されない操作
EROFS	書き込み不可のファイルシステム
EXDEV	同一ファイルシステムでない
ENOENT	ファイルまたはディレクトリがない
ENOSPC	ディスクが容量不足
ENODEV	デバイスに対する許されない操作
EIO	入出力エラー
EDEADLK	ロック異常
EBUSY	ビジー状態
＜ 0	その他のエラー

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.6.12 システムコール

ファイルシステムの接続

tkse_attach

C言語インタフェース

```
ER ercd = tkse_attach( const TC *devnm, const char *connm, int mode );
```

パラメータ

const TC	*devnm	デバイス名
const char	*connm	接続名
int	mode	接続モード (SF_STDFS ・ SF_FATFS ・ SF_CDROM) | [SF_RONLY] SF_RONLY 0x0001 読み込みのみ SF_STDFS 0x0000 T-Kernel 標準ファイルシステム SF_FATFS 0x0100 FAT ファイルシステム SF_CDROM 0x0200 CD-ROM ファイルシステム

リターンパラメータ

ER	ercd	= 0 正常終了 < 0 エラーコード

解説

デバイス名 devnm のデバイスのファイルシステムを、接続名 connm、接続モード mode で接続する。

接続名は、最大 16 バイト(文字列終端の '¥0' を除く)。ただし、T-Kernel 標準ファイルシステムの場合は、最大 8 文字(1 バイト文字のみであれば最大 8 バイト、2 バイト文字のみであれば最大 16 バイト)となる。

接続モード mode は、接続するファイルシステムの種類を指定する。

SF_STDFS/SF_FATFS/SF_CDROM を指定することにより、T-Kernel 標準ファイルシステム/FAT ファイルシステム/CD-ROM ファイルシステムをそれぞれ接続する。

ファイルシステムの種類に加えて mode に SF_RONLY を指定した場合は、ファイルシステムを読み込み専用として接続する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステムの切断

tkse_detach

C言語インタフェース

```
ER tkse_detach( const TC *devnm, int eject );
```

パラメータ

const TC	*devnm	デバイス名
int	eject	イジェクト指定
		＝0　イジェクトしない
		≠0　イジェクトする(イジェクト不可能なデバイスの時は無視)

リターンパラメータ

ER	ercd	＝ 0　正常終了
		＜ 0　エラーコード

解説

デバイス名 devnm のデバイスのファイルシステムを切断する。
ファイルシステム上のファイルをオープンしている場合は切断できない。
ディスクキャッシュ上にデータが存在する場合はすべて同期される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル／ディレクトリのオープン

tkse_open

C言語インタフェース

```
ER ercd = tkse_open( const char *path, int oflag, mode_t mode );
```

パラメータ

const char	*path	オープンするファイルパス
int	oflag	ファイルまたはディレクトリのオープンモード
mode_t	mode	O_CREAT 指定時モード

リターンパラメータ

ER	ercd	≧ 0	正常終了（ファイルディスクリプタ）
		＜ 0	エラーコード

解説

パス名 path のファイルまたはディレクトリをオープンモード oflag でオープンする。
リターンパラメータにファイルディスクリプタ(≧0)を返す。
ファイルディスクリプタは使用されていない番号の内、最小のものとなる。

```
oflag :＝ (O_RDONLY ・ O_WRONLY ・ O_RDWR) | [O_CREAT | [O_EXCL]] | [O_TRUNC] | [O_APPEND]
```

oflag は以下の値のいずれかを指定する。

```
O_RDONLY  0x0000 読み込みのみ
O_WRONLY  0x0001 書き込みのみ
O_RDWR    0x0002 読み込み書き込み
```

また oflag は、オプションとして以下の値の論理和を追加で指定できる。

```
O_CREAT   0x0200   ファイルがなければファイル生成
O_TRUNC   0x0400   ファイル内容削除
O_EXCL    0x0800   ファイルがあればエラー
O_APPEND  0x0008   常に末尾に追加
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

O_CREAT　ファイルがなければファイルを生成する。ファイルがすでにある場合には
このフラグは何の効果もない。
mode で指定されたモードでファイルを生成する。

O_EXCL　O_CREAT と合わせて指定する。ファイルがすでにある場合は、エラーとする。
O_CREAT が指定されていなければ無視する。

O_TRUNC　ファイルの内容を破棄し、ファイルサイズを0にする。
ディレクトリに対して指定した場合は無視される。
読み込み専用オープン（ O_RDONLY 指定 ）の時は無視される。

O_APPEND　ファイルへの書き込み時に、常にファイルの末尾に追加する。
このとき、現在位置を末尾に移動する。
tkse_write の直前に tkse_lseek でファイルの末尾に移動するのと同じである。

mode は O_CREAT を指定した場合にのみ指定する必要がある。
mode は以下の値の論理和とする。

S_IRWXU 00700　所有者の RWX マスク
S_IRUSR 00400　所有者の R 読み込み許可
S_IWUSR 00200　所有者の W 書き込み許可
S_IXUSR 00100　所有者の X 実行許可
S_IRWXG 00070　グループの RWX マスク
S_IRGRP 00040　グループの R 読み込み許可
S_IWGRP 00020　グループの W 書き込み許可
S_IXGRP 00010　グループの X 実行許可
S_IRWXO 00007　その他の RWX マスク
S_IROTH 00004　その他の R 読み込み許可
S_IWOTH 00002　その他の W 書き込み許可
S_IXOTH 00001　その他の X 実行許可
S_ISUID 04000　実行時ユーザ ID 設定
S_ISGID 02000　実行時グループ ID 設定
S_ISVTX 01000　スティッキービット

これらの mode 指定は、対象となるファイルシステムによってその有効範囲が異なる。対象ファイルシステムにおいて無効な指定は無視される。
なお、現在は umask の機能が実装されていないため、umask によるマスクは行われない。

・T-Kernel 標準ファイルシステム
tkse_open() 時に決定したファイルタイプは tkse_close() されるまで変わることはない。例えば、tkse_open()にリンクレコードを含むファイルをディレクトリとしてオープンした場合、tkse_close() される

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

までに他プロセスがリンクレコードをすべて削除したとしてもファイルタイプはディレクトリとして保たれる。

所有者・グループ・その他のすべての書き込みが不許可だった場合に、ファイルアクセス属性を書き込み不可に設定する。それ以外の場合は書き込み許可となる。

ファイルのアクセスモードは、常にデフォルトのアクセスモードが設定される。

・FAT ファイルシステム

所有者・グループ・その他のすべての書き込みが不許可だった場合に、リードオンリー属性となる。

それ以外の場合は書き込み許可となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル／ディレクトリのクローズ

tkse_close

C言語インタフェース

```
ER ercd = tkse_close( int fildes );
```

パラメータ

int	fildes	ファイルディスクリプタ	

リターンパラメータ

ER	ercd	＝ 0	正常終了
		＜ 0	エラーコード

解説

ファイルディスクリプタ fildes で示される、オープン中のファイル／ディレクトリをクローズする。

書き込みオープンしたファイルをクローズする場合、同じファイルを他で同時に書き込みオープンしていなければ、ファイルのディスクキャッシュ同期処理（変更内容をディスクキャッシュからディスクに書き戻す処理）を行う。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル／ディレクトリの現在位置の移動

tkse_lseek

C言語インタフェース

```
ER ercd = tkse_lseek( int fildes, off_t offset, int whence );
```

パラメータ

int	fildes	ファイルディスクリプタ
off_t	offset	指定位置からのオフセット
int	whence	起点指定
	(SEEK_SET ・ SEEK_CUR ・ SEEK_END)	
	SEEK_SET 0	offset の位置に移動
	SEEK_CUR 1	現在位置 + offset に移動
	SEEK_END 2	終端 + offset に移動

リターンパラメータ

ER	ercd	≧ 0	正常終了（移動後の現在位置）
		＜ 0	エラーコード

解説

ファイルディスクリプタ fildes で示されるファイル／ディレクトリの現在位置（バイト単位の位置）を移動する。

fildes がディレクトリの場合、現在位置を先頭に戻す（0 にセットする）目的以外で tkse_lseek を使用するべきではない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルの読み込み

tkse_read

C言語インタフェース

```
ER ercd = tkse_read( int fildes, void *buf, size_t nbyte );
```

パラメータ

int	fildes	ファイルディスクリプタ
void	*buf	リードバッファ
size_t	nbyte	リードサイズ(バイト単位)

リターンパラメータ

ER	ercd	≧ 0	正常終了（読み込みに成功したバイト数）
		＜ 0	エラーコード

解説

ファイルディスクリプタ fildes で示されるファイルの現在位置から、nbyte バイトを buf に読み込む。また、読み込んだバイト数分、ファイルの現在位置を進める。

リターンパラメータとして読み込みに成功したバイト数を返す。ファイルの現在位置がファイル終端であった場合は 0 を返す。読み込み途中でエラーが発生した場合はエラーコードを返す。この場合、ファイルの現在位置は変更されない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルの書き込み

tkse_write

C言語インタフェース

```
ER ercd = tkse_write( int fildes, const void *buf, size_t nbyte );
```

パラメータ

const int	fildes	ファイルディスクリプタ
void	*buf	ライトバッファ
size_t	nbyte	ライトサイズ（バイト数）

リターンパラメータ

ER	ercd	≧ 0	正常終了（書き込みに成功したバイト数）
		＜ 0	エラーコード

解説

ファイルディスクリプタ fildes で示されるファイルの現在位置に、buf の内容を nbyte 書き込む。また、書き込んだバイト数分、ファイルの現在位置を進める。

リターンパラメータとして書き込みに成功したバイト数を返す。書き込み途中でエラーが発生した場合はエラーコードを返す。この場合、どこまでデータが書き込まれたかは不確定である。また、ファイルの現在位置は変更されない。

ファイルの現在位置が実際のファイルサイズを超えている場合は、以下の動作となる。

・T-Kernel 標準ファイルシステム
エラーとなる。

・FAT ファイルシステム
ファイルの終端から現在位置まで、ゼロのデータが書き込まれる。
リターンパラメータのバイト数にこの領域は含まれない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ディレクトリエントリの取り出し

tkse_getdents

C言語インタフェース

```
ER ercd = tkse_getdents( int fildes, struct dirent *buf, size_t nbyte );
```

パラメータ

int	fildes	ファイルディスクリプタ
struct dirent	*buf	ディレクトリエントリの読み込みバッファ
size_t	nbyte	読み込みサイズ（バイト）

リターンパラメータ

ER	ercd	≧ 0	正常終了（読み込みに成功したバイト数）
		＜ 0	エラーコード

解説

ファイルディスクリプタ fildes のディレクトリの現在位置からディレクトリエントリ(レコード)を読み込み、struct dirent の形式で buf に書き込む。

nbyte は buf で指定した領域のサイズをバイト数で指定する。指定したサイズに格納できる数だけ、連続してディレクトリエントリを読み込む。読み込み成功後、ディレクトリの現在位置を、最後に読み込んだディレクトリエントリの次のディレクトリエントリを指すように移動する。

```
struct dirent {
    unsigned int     d_fileno;      /* ファイル番号 */
    unsigned short   d_reclen;      /* レコード長(バイト数) */
    unsigned char    d_type;        /* ファイルタイプ */
    unsigned char    d_namlen;      /* d_name の文字列長 */
    char             d_name[255+1]; /* ファイル名 */
};
```

ファイルタイプ:

DT_UNKNOWN	0	不明
DT_FIFO	1	名前付きパイプ(FIFO)
DT_CHR	2	キャラクタ型特殊ファイル
DT_DIR	4	ディレクトリ
DT_BLK	6	ブロック型特殊ファイル

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

DT_REG	8 普通のファイル
DT_LNK	10 シンボリックリンク
DT_SOCK	12 ソケット

struct dirent は可変長データで、d_reclen によってそのサイズを知ることができる。複数のディレクトリエントリを読み込んだ場合、buf 上に読み込まれた現在のディレクトリエントリの先頭アドレスを d_reclen だけ進めることによって、次のディレクトリエントリの格納位置を知ることができる。

リターンパラメータとして読み込みに成功したバイト数を返す。現在位置がディレクトリエントリの終端であった場合は 0 を返す。読み込み途中でエラーが発生した場合はエラーコードを返す。この場合、現在位置は変更されない。

・T-Kernel 標準ファイルシステム

d_name は出現順を含むファイル名が格納される。また、ディレクトリエントリの読み込みを複数回に分けて行う場合、読み込み中に他プロセスがファイル操作を行うことで、ファイルの出現順が正しい値でなくなる可能性がある。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル情報の取得

tkse_stat

C言語インタフェース

```
ER ercd = tkse_stat( const char *path, struct stat *sb );
```

パラメータ

const char	*path	ファイルパス名
struct stat	*sb	ファイル情報取得バッファ

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

パス名 path のファイルの情報を取得し sb に格納する。

```
struct stat {
    dev_t           st_dev;       /* デバイス ID */
    ino_t           st_ino;       /* ファイル通し番号 */
    mode_t          st_mode;      /* ファイルモード */
    nlink_t         st_nlink;     /* リンクの数 */
    uid_t           st_uid;       /* 所有者 ID */
    gid_t           st_gid;       /* グループ ID */
    dev_t           st_rdev;      /* デバイスタイプ */
    struct timespec st_atimespec; /* 最新アクセス時刻 */
    struct timespec st_mtimespec; /* 最新更新時刻 */
    struct timespec st_ctimespec; /* 最新ファイル状態更新時刻 */
    off_t           st_size;      /* ファイルの大きさ(バイト数) * /
    int64_t         st_blocks;    /* ファイルの割り当てブロック数 */
    u_int32_t       st_blksize;   /* ブロックサイズ(バイト数) */
    u_int32_t       st_flags;     /* ユーザ定義フラグ */
    u_int32_t       st_gen;       /* ファイル生成番号 */
    int32_t         st_lspare;    /* (予約) */
    int64_t         st_qspare[2]; /* (予約) */
};
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
#define st_atime     st_atimespec.tv_sec
#define st_mtime     st_mtimespec.tv_sec
#define st_ctime     st_ctimespec.tv_sec

#define S_BLKSIZE    512  /* st_blocks の単位となるブロックのサイズ(バイト数) */
```

ファイルモード st_mode は以下の値の論理和を返す。

```
#define S_IRWXU      0000700  /* 所有者の RWX マスク */
#define S_IRUSR      0000400  /* 所有者の R 読み込み許可 */
#define S_IWUSR      0000200  /* 所有者の W 書き込み許可 */
#define S_IXUSR      0000100  /* 所有者の X 実行許可 */
#define S_IRWXG      0000070  /* グループの RWX マスク */
#define S_IRGRP      0000040  /* グループの R 読み込み許可 */
#define S_IWGRP      0000020  /* グループの W 書き込み許可 */
#define S_IXGRP      0000010  /* グループの X 実行許可 */
#define S_IRWXO      0000007  /* その他の RWX マスク */
#define S_IROTH      0000004  /* その他の R 読み込み許可 */
#define S_IWOTH      0000002  /* その他の W 書き込み許可 */
#define S_IXOTH      0000001  /* その他の X 実行許可 */
#define S_ISUID      0004000  /* 実行時ユーザ ID 設定 */
#define S_ISGID      0002000  /* 実行時グループ ID 設定 */
#define S_ISVTX      0001000  /* スティッキービット */
#define S_IFMT       0170000  /* ファイルタイプのマスク */
#define S_IFIFO      0010000  /* 名前付きパイプ(FIFO)  */
#define S_IFCHR      0020000  /* キャラクタ型特殊ファイル */
#define S_IFDIR      0040000  /* ディレクトリ */
#define S_IFBLK      0060000  /* ブロック型特殊ファイル */
#define S_IFREG      0100000  /* 通常ファイル */
#define S_IFLNK      0120000  /* シンボリックリンク */
#define S_IFSOCK     0140000  /* ソケット */
```

ユーザ定義フラグ st_flags は以下の値の論理和を返す。

```
#define SF_ARCHIVED  0x00010000   /* アーカイブファイル */
#define SF_SYSTEM    0x40000000   /* システムファイル */
#define SF_HIDDEN    0x80000000   /* 隠しファイル */
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
struct timespec {
    time_t tv_sec;     /* 秒 */
    long   tv_nsec;    /* ナノ秒 */
};
```

time_t は 1985 年 1 月 1 日 00:00:00 GMT を基準日時とした秒数となる（TRON 仕様に基づいている）。

ファイルに記録されている時刻が基準日時より過去の時刻の場合は 0 (tv_sec＝0, tv_nsec＝0) を返す。また、ファイルに記録されている時刻が time_t で表せる範囲を越えた未来の時刻の場合は 0x7fffffff(tv_sec＝0x7fffffff, tv_nsec＝0)を返す。

ファイルに記録される時刻の更新は、そのファイルシステムによって規定されたタイミングで行われる。

以下の各要素については、対象とするファイルシステムによって取得できる情報が異なる。

・T-Kernel 標準ファイルシステム

st_dev デバイス ID

デバイス ID はデバイス登録時に動的に割り当てられるものであるため、固定的な値ではない。

st_ino ファイル ID

st_mode S_IRUSR S_IRGRP S_IROTH は、常に設定される。

S_IXUSR S_IXGRP S_IXOTH は、常に設定される。

S_IWUSR S_IWGRP S_IWOTH は、書き込み不可属性が設定されていない場合のみ設定される。

所有者、グループ、その他で独立した属性は設定されない。常に同じになる。

ファイルタイプは、S_IFDIR S_IFREG のいずれかが設定される。

tkse_fstat では、オープン時のファイルタイプが設定され、それ以外は現在のファイルタイプ、またはリンクレコードの有無で決定される。

その他の属性は設定されることはない。

st_nlink	ファイル参照数
st_uid	(常に 0)
st_gid	(常に 0)
st_rdev	(常に 0)
st_atimespec	最新アクセス時刻
st_mtimespec	最新更新時刻
st_ctimespec	ファイル生成時刻
st_size	対象レコードサイズ

アクセスの対象となるレコードのみのサイズとなるため、複数のデータレコードが含まれる場合、ファイルサイズより小さくなる。

対象ファイルのファイルタイプがディレクトリの場合はリンクレコード数が設定される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

st_blocks	総使用ブロック数 全レコードおよび管理情報を含んだ使用ブロック数となる。
st_blksize	論理ブロックサイズ
st_flags	隠蔽仮身の場合に SF_HIDDEN が設定される。
st_gen	(常に 0)

・FAT ファイルシステム

st_dev デバイス ID

デバイス ID はデバイス登録時に動的に割り当てられるものであるため、固定的な値ではない。

st_ino	ディレクトリエントリのディスク上の位置を元にした値 必ずしも固定的な値ではない。
st_mode	S_IRUSR S_IRGRP S_IROTH は、常に設定される。 S_IXUSR S_IXGRP S_IXOTH は、常に設定される。 S_IWUSR S_IWGRP S_IWOTH は、書き込み不可属性が設定されていない場合のみ設定される。 所有者、グループ、その他で独立した属性は設定されない。常に同じになる。 ファイルタイプは、S_IFDIR S_IFREG のいずれかが設定される。 その他の属性は設定されることはない。
st_nlink	(常に 1)
st_uid	(常に 0)
st_gid	(常に 0)
st_rdev	(常に 0)
st_atimespec	最新アクセス日(時刻は常に 00:00:00)
st_mtimespec	最新更新時刻
st_ctimespec	ファイル生成時刻 アクセス時刻と生成時刻は VFAT の場合のみとなる。VFAT でない場合は、すべて更新時刻が設定される。
st_size	ファイルサイズ
st_blocks	使用ブロック数
st_blksize	クラスタサイズ
st_flags	FAT のファイルタイプにしたがって、SF_ARCHIVED SF_SYSTEM SF_HIDDEN が設定される。
st_gen	(常に 0)

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

・CD-ROM ファイルシステム

st_dev	デバイス ID

デバイス ID はデバイス登録時に動的に割り当てられるものであるため、固定的 な値ではない。

st_ino	ディレクトリレコードのディスク上の位置を元にした値
st_mode	S_IRUSR S_IRGRP S_IROTH は、常に設定される。 S_IXUSR S_IXGRP S_IXOTH は、常に設定される。 ファイルタイプは、S_IFDIR S_IFREG のいずれかが設定される。 その他の属性は設定されることはない。
st_nlink	(常に 1)
st_uid	(常に 0)
st_gid	(常に 0)
st_rdev	(常に 0)
st_atimespec	記録日時
st_mtimespec	記録日時
st_ctimespec	記録日時
st_size	ファイルサイズ
st_blocks	使用ブロック数
st_blksize	論理ブロックサイズ
st_flags	隠しファイルのとき SF_HIDDEN が設定される。
st_gen	(常に 1)

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル情報の取得

tkse_lstat

C言語インタフェース

```
ER ercd = tkse_lstat( const char *path, struct stat *sb );
```

パラメータ

const char	*path	ファイルパス名
struct stat	*sb	ファイル情報取得バッファ

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

パス名 path のファイルの情報を取得し sb に格納する。
標準入出力機能はシンボリックリンクをサポートしないため、tkse_lstat()は tkse_stat()と同じ動作となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル情報の取得

tkse_fstat

C言語インタフェース

```
ER ercd = tkse_fstat( int fildes, struct stat *sb );
```

パラメータ

int	fildes	ファイルディスクリプタ
struct stat	*sb	ファイル情報取得バッファ

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

ファイルディスクリプタ filedes で示されるファイルの情報を取得し sb に格納する。
通常のファイルのみでなく、コンソール I/O（標準入出力）のディスクリプタも指定可能である。

・コンソール I/O（標準入出力）

st_dev	(常に 0)
st_ino	(常に 0)
st_mode	S_IRUSR|S_IWUSR|S_IFCHR が設定される。(固定値)
st_nlink	(常に 1)
st_uid	(常に 0)
st_gid	(常に 0)
st_rdev	(常に 0)
st_atimespec	(常に 0)
st_mtimespec	(常に 0)
st_ctimespec	(常に 0)
st_size	(常に 0)
st_blocks	(常に 0)
st_blksize	(常に 0)
st_flags	(常に 0)
st_gen	(常に 0)

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル名の変更

tkse_rename

C言語インタフェース

```
ER ercd = tkse_rename( const char *from, const char *to );
```

パラメータ

const char	*from	変更前ファイル名
const char	*to	変更後ファイル名

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

ファイル名 from をファイル名 to に変更する。

to がすでに存在する場合は、to を削除する。このとき from と to は同じタイプ（両方ともファイル、または両方ともディレクトリ）でなければならない。

from と to が異なるディレクトリ上にあるときはディレクトリの移動となる。

from と to は同じファイルシステム上に存在しなければならない。異なるファイルシステム上に存在する場合はエラーとなる。

・T-Kernel 標準ファイルシステム

to が指すファイルに書き込み不可属性が設定されている場合はエラーとなる。

パス名 to が指すファイルは未オープン状態でなければならない。

ディレクトリの rename で、パス名 to が from を含んでいる場合や、to が from のサブディレクトリの場合でもリネームを行う。

※パス名 to が from を含んでいる場合、以後パス名によるファイルのアクセスが不可能になる場合がある。

・FAT ファイルシステム

to は未オープン状態でなければならない。

ディレクトリの rename の場合、パス名 to が from を含んでいる場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ディレクトリエントリの削除

tkse_unlink

C言語インタフェース

```
ER ercd = tkse_unlink( const char *path );
```

パラメータ

const char	*path	削除するディレクトリパス

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

パス名 path のファイルを削除する。
ディレクトリ、およびオープン中のファイルは削除できない。

・T-Kernel 標準ファイルシステム
　ファイルタイプが 6 以外の場合、呼び出された時点のリンクレコードの有無でディレクトリかどうかを判別する。従って、ファイルタイプが 6 であるか、リンクレコードを含むファイルである場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ディレクトリの作成

tkse_mkdir

C言語インタフェース

```
ER ercd = tkse_mkdir( const char *path, mode_t mode );
```

パラメータ

const char	*path	作成するディレクトリ名
mode_t	mode	ディレクトリ作成モード ※ tkse_open() の mode と同じ

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

パス名 path のディレクトリを作成する。
ディレクトリ名として ″.″ と ″..″は使用できない。使用した場合はエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ディレクトリの削除

tkse_rmdir

C言語インタフェース

```
ER ercd = tkse_rmdir( const char *path );
```

パラメータ

const char	*path	削除するディレクトリ名

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

パス名 path で指定されたディレクトリを削除する。
削除するディレクトリは空（ "." および ".." を除く）でなければならない。
ディレクトリ "." と ".." は削除できない。またオープン状態のディレクトリは削除できない。

・T-Kernel 標準ファイルシステム
呼び出された時点のリンクレコードの有無でファイルタイプを判別する。
従って、リンクレコードを含まないファイルは、ファイルタイプに関わらず削除対象となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルディスクリプタの複製

tkse_dup

C言語インタフェース

```
ER ercd = tkse_dup( int oldd );
```

パラメータ

int	oldd	複製するファイルディスクリプタ	

リターンパラメータ

ER	ercd	≧ 0	正常終了(複製したファイルディスクリプタ)
		＜ 0	エラーコード

解説

ファイルディスクリプタ oldd を複製し、新しいファイルディスクリプタを返す。
ファイルディスクリプタは使用されていない番号の内、最小のものとなる。
複製したファイルディスクリプタは oldd と同じものとして扱われ、現在位置ポインタを共有する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルディスクリプタの複製

tkse_dup2

C言語インタフェース

```
ER ercd = tkse_dup2( int oldd, int newd );
```

パラメータ

int	oldd		複製するファイルディスクリプタ
int	newd		新しいファイルディスクリプタ

リターンパラメータ

ER	ercd	≧ 0	正常終了(複製したファイルディスクリプタ)
		＜ 0	エラーコード

解説

ファイルディスクリプタ oldd を、新しいファイルディスクリプタ newd として複製する。
複製したファイルディスクリプタ newd は oldd と同じものとして扱われ、現在位置ポインタを共有する。
newd がすでに使用されている場合は、そのファイルをクローズしてから複製を行う。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルのディスクキャッシュ内容とディスクの同期

tkse_fsync

C言語インタフェース

```
ER ercd = tkse_fsync( int fildes );
```

パラメータ

int	fildes	ファイルディスクリプタ	

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

ファイルディスクリプタ fildes に対する書き込み操作のうち、ディスクキャッシュからディスクに書き戻されていない（同期されていない）データを書き戻す。

ディスクへの書き込みが終了した後にリターンする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

カレントディレクトリの変更

tkse_chdir

C言語インタフェース

```
ER ercd = tkse_chdir( const B *path );
```

パラメータ

const B	*path	変更するディレクトリのパス

リターンパラメータ

ER	ercd	= 0　正常終了
		< 0　エラーコード

解説

カレントディレクトリ（作業ディレクトリ）を、パス名 path が指すディレクトリに変更する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

カレントディレクトリの変更

tkse_fchdir

C言語インタフェース

```
ER ercd = tkse_fchdir( int fildes );
```

パラメータ

int	fildes	変更するディレクトリのファイルディスクリプタ

リターンパラメータ

ER	ercd	＝ 0　正常終了
		＜ 0　エラーコード

解説

カレントディレクトリ（作業ディレクトリ）を、ファイルディスクリプタ fildes としてオープンされているディレクトリに変更する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルモードの変更

tkse_chmod

C言語インタフェース

```
#include <extension/seio.h>

ER tkse_chmod( const B *path, mode_t mode );
```

パラメータ

const B	*path	ファイルまたはディレクトリのパス
mode_t	mode	モード指定

リターンパラメータ

ER	ercd	＝ 0	正常終了
		＜ 0	エラーコード

解説

パス名 path で指定したファイルまたはディレクトリのモードを変更する。
mode は以下の値の論理和とする。

S_IRWXU	00700	所有者の RWX マスク
S_IRUSR	00400	所有者の R 読み込み許可
S_IWUSR	00200	所有者の W 書き込み許可
S_IXUSR	00100	所有者の X 実行許可
S_IRWXG	00070	グループの RWX マスク
S_IRGRP	00040	グループの R 読み込み許可
S_IWGRP	00020	グループの W 書き込み許可
S_IXGRP	00010	グループの X 実行許可
S_IRWXO	00007	その他の RWX マスク
S_IROTH	00004	その他の R 読み込み許可
S_IWOTH	00002	その他の W 書き込み許可
S_IXOTH	00001	その他の X 実行許可
S_ISUID	04000	実行時ユーザ ID 設定
S_ISGID	02000	実行時グループ ID 設定
S_ISVTX	01000	スティッキービット

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

mode 指定は、対象となるファイルシステムによってその有効範囲が異なる。対象ファイルシステムにおいて無効な指定は無視される。

すでにオープンされているファイルのファイルモードを変更した場合、その変更はクローズされるまで影響を与えない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルモードの変更

tkse_fchmod

C言語インタフェース

```
ER ercd = tkse_fchmod( int fildes, mode_t mode );
```

パラメータ

int	fildes	ファイルディスクリプタ
mode_t	mode	モード指定

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

ファイルディスクリプタ fildes としてオープンされているファイルまたはディレクトリのモードを変更する。mode 指定は、対象となるファイルシステムによってその有効範囲が異なる。対象ファイルシステムにおいて無効な指定は無視される。

すでにオープンされているファイルのファイルモードを変更した場合、その変更はクローズされるまで影響を与えない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルの作成

tkse_creat

C言語インタフェース

```
ER ercd = tkse_creat( const B *path, mode_t mode );
```

パラメータ

const B	*path		パス名
mode_t	mode		モード指定

リターンパラメータ

ER	ercd	≧ 0	正常終了（ファイルディスクリプタ）
		＜ 0	エラーコード

解説

パス名 path のファイルを生成してオープンする。
戻値のファイルディスクリプタは使用されていない番号の内、最小のものとなる。
tkse_open()の oflag に(O_CREAT | O_WRONLY | O_TRUNC)を指定した場合と同等の処理を行う。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

アクセス時間／修正時間の変更

tkse_utimes

C言語インタフェース

```
ER ercd = tkse_utimes( const B *path, const struct timeval times[2] );
```

パラメータ

const B	*path	パス名
const struct timeval	times[2]	アクセス時間・修正時間

リターンパラメータ

ER	ercd	＝ 0 正常終了 ＜ 0 エラーコード

解説

パス名 path が示すファイルまたはディレクトリのアクセス時間、修正時間を変更する。
アクセス時間を times[0].tv_sec に、また修正時間を times[1].tv_sec に設定する。
times は 1985 年 1 月 1 日 00:00:00 GMT を基準日時とした秒数を設定する。これは TRON 仕様に基づく。
times ＝ NULL の場合は、アクセス時間と修正時間を現在時刻に設定する。

・標準 ファイルシステム
　アクセス時間、修正時間に 0 を指定した場合は、その時間を変更しない。

```
struct timeval {
    long tv_sec;      /* 秒 */
    long tv_usec;     /* マイクロ秒 */
};
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル作成マスクの設定

tkse_umask

C言語インタフェース

```
mode_t tkse_umask( mode_t cmask );
```

パラメータ

mode_t	cmask	ファイル生成マスク値

リターンパラメータ

ER	ercd	≧ 0	正常終了（設定前のマスク値）
		＜ 0	エラーコード

解説

自プロセスのファイル生成マスク値を cmask に設定する。

ファイル作成時に指定したモード値 mode に、マスク値 cmask で指定した値を取り除いたものが、ファイル生成時に適用されるモード値となる。

cmask は以下の値の論理和である。

S_IRWXU	00700	所有者の RWX マスク
S_IRUSR	00400	所有者の R 読み込み許可
S_IWUSR	00200	所有者の W 書き込み許可
S_IXUSR	00100	所有者の X 実行許可
S_IRWXG	00070	グループの RWX マスク
S_IRGRP	00040	グループの R 読み込み許可
S_IWGRP	00020	グループの W 書き込み許可
S_IXGRP	00010	グループの X 実行許可
S_IRWXO	00007	その他の RWX マスク
S_IROTH	00004	その他の R 読み込み許可
S_IWOTH	00002	その他の W 書き込み許可
S_IXOTH	00001	その他の X 実行許可
S_ISUID	04000	実行時ユーザ ID 設定
S_ISGID	02000	実行時グループ ID 設定
S_ISVTX	01000	スティッキービット

これらの umask 指定は、対象となるファイルシステムによってその有効範囲が異なる。ファイルシステム

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

がサポートしていない指定は無視される。

プロセスは親プロセスの cmask を引き継ぐ。初期プロセスの cmask は 0 とする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルサイズを指定長に設定

tkse_truncate

C言語インタフェース

```
ER ercd = tkse_truncate( const B *path, off_t length );
```

パラメータ

const B	*path	パス名
off_t	length	ファイルサイズ

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

パス名 path で指定したファイルのファイルサイズを、length バイトに延長、もしくは切り詰める。

length がファイルサイズより小さい場合、ファイルサイズは length バイトに切り詰められる。切り詰められた部分のデータは失われる。

length がファイルサイズより大きい場合、ファイルサイズは length バイトに延長される。延長された部分には 0 が書き込まれる。

length がファイルサイズと同じ場合は何も処理しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルサイズを指定長に設定

tkse_ftruncate

C言語インタフェース

```
ER ercd = tkse_ftruncate( int fildes, off_t length );
```

パラメータ

int	fildes	ファイルディスクリプタ
off_t	length	ファイルサイズ

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

ファイルディスクリプタ fildes としてオープンされているファイルのファイルサイズを、length バイトに延長、もしくは切り詰める。

length がファイルサイズより小さい場合、ファイルサイズは length バイトに切り詰められる。切り詰められた部分のデータは失われる。

length がファイルサイズより大きい場合、ファイルサイズは length バイトに延長される。延長された部分には 0 が書き込まれる。

length がファイルサイズと同じ場合は何も処理しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ディスクキャッシュの内容とディスクの同期

tkse_sync

C言語インタフェース

```
void tkse_sync( void );
```

パラメータ

なし

リターンパラメータ

なし

解説

ディスクキャッシュからディスクに書き戻されていない（同期されていない）すべてのデータを書き戻す。
ディスクへの書き込みが終了した後にリターンする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステムのリストの取得

tkse_getfsstat

C言語インタフェース

```
ER ercd = tkse_getfsstat( struct statfs *buf, W bufsize, int flags );
```

パラメータ

struct statfs *buf		ファイルシステム情報の格納領域
W	bufsize	格納領域のサイズ
int	flags	フラグ（未使用）

リターンパラメータ

ER	ercd	≧ 0	正常終了(取得したファイルシステム情報の個数)
		＜ 0	エラーコード

解説

接続されているすべてのファイルシステムに関する情報を取得して buf に格納する。

bufsize は buf の領域サイズ（バイト数）を指定する。ファイルシステム情報のサイズが bufsize を超えていた場合は、buf に格納可能な個数までのファイルシステム情報を取得する。例として、bufsize が sizeof(struct statfs) * 10 である場合は、最大 10 個までのファイルシステム情報を格納可能である。

戻値として、取得に成功したファイルシステム情報の個数を返す。buf に NULL を指定した場合、bufsize は無視され、現在接続しているファイルシステムの個数のみを返す。

flags は将来拡張のために予約された引数である。常に MNT_WAIT （＝ 1)を指定する。

ルートファイルシステムは仮想的なファイルシステムであるため、ファイルシステム情報は取得できない。

```
typedef struct fsid {
    W val[2];
} fsid_t;                       /* ファイルシステム ID */

#define MNAMELEN 90 /* 接続名／デバイス名の最大長 */

struct statfs {
    W       f_spare2;           /* (空き) */
    W       f_bsize;            /* 論理ブロックサイズ(B:バイト数) */
    W       f_iosize;           /* 最適な転送ブロックサイズ(B) */
    W       f_blocks;           /* ファイルシステム容量(LB:論理ブロック数) */
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
    W       f_bfree;            /* ファイルシステム空き容量(LB) */
    W       f_bavail;           /* 一般ユーザが利用可能な空き容量(LB) */
    W       f_files;            /* ※最大ファイル数 */
    W       f_ffree;            /* ※空きファイル数 */
    fsid_t  f_fsid;             /* ファイルシステム ID (常に 0) */
    uid_t   f_owner;            /* 接続したユーザ (常に 0) */
    int     f_type;             /* ファイルシステムタイプ */
    int     f_flags;            /* 接続フラグ */
    W       f_spare[6];         /* (空き) */
    B       f_mntonname[MNAMELEN];      /* 接続名 */
    B       f_mntfromname[MNAMELEN];    /* デバイス名 */
};
```

※印の項目は、ファイルシステムによっては値が定義されない場合があり、その場合 -1 が設定される。

ユーザ単位のクオータ制御は行わないため、一般ユーザが利用可能な空き容量 f_bavail は、ファイルシステムの空き容量 f_bfree と等しい。

ファイルシステムタイプ f_type は以下のいずれかの値となる。

```
#define MOUNT_FATFS   4   /* FAT ファイルシステム */
#define MOUNT_CD9660  14  /* CD-ROM ファイルシステム */
#define MOUNT_STDFS   20  /* 標準 ファイルシステム */
```

接続フラグ f_flags は以下の値の論理和となる。

```
#define MNT_RDONLY 0x00000001 /* 読み込み専用 */
```

接続名 f_mntonname はルートからのパス名となる。

(例) "/SYS"

デバイス名 f_mntfromname は、デバイス名の前に "/dev/" を付けた名称となる。

(例) "/dev/pca0"

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

標準ファイルのLINKの取得

tkse_getlink

C言語インタフェース

```
ER ercd = tkse_getlink( const B *path, B *buf );
```

パラメータ

const B	*path	標準入出力のファイルパス
B	*buf	LINK情報の格納領域

リターンパラメータ

ER	ercd	= 0	正常終了
		< 0	エラーコード

解説

パス名pathで指定したファイルまたはディレクトリの、標準ファイルシステム仕様に基づくLINK情報をbufに返す。

bufはsizeof(LINK)以上の大きさを持つ領域でなければならない。

pathのファイルまたはディレクトリが標準ファイルシステム上のファイルである場合のみ、bufにLINK情報を格納する。標準ファイルシステム上のファイルでない場合は、bufの内容は不定となりエラーを返す。

対象のファイルまたはディレクトリへのアクセス権がなくてもLINK情報は取得可能である。ただし、pathに含まれる対象のファイルまたはディレクトリに達するまでの経路上のディレクトリには、tkse_open()でファイルをオープンするときと同様の、ディレクトリに対するアクセス権を必要とする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.7 標準ファイル管理

4.7.1 概要

標準ファイル管理機能は、Standard Extension の標準ファイルシステム、およびそのファイルを操作するための機能を提供する。

通常の用途でファイル操作を行う場合は Standard Extension の標準入出力機能を使用することを推奨する。標準ファイルシステム独自の機能を使用する場合に、標準ファイル管理機能を直接使用する。

標準ファイルシステムは、実身／仮身モデルを基本とした構造となっており、以下のような特徴を持つ。

・可変長レコードの順序列によるファイル編成（ レコードストリーム ）
・ファイルに含まれるリンク（ 仮身 ）による任意のネットワーク状の参照関係
（ 従来のファイルシステムにおけるディレクトリは存在しない ）
・リンク（ 仮身 ）によるファイルの直接的なアクセス

ファイルは時系列上において複数のユーザにより使用され、さらにネットワーク環境では複数のユーザにより同時に使用されるためファイルのアクセス管理をきめ細かく行い、高いレベルの保護機構を提供している。ただし Standard Extension の現バージョンでは、複数ユーザおよびネットワーク環境の対応は行わない。

4.7.2 ファイルとリンク

1 つのファイルは可変長の順序付けられたレコードの列から構成される。

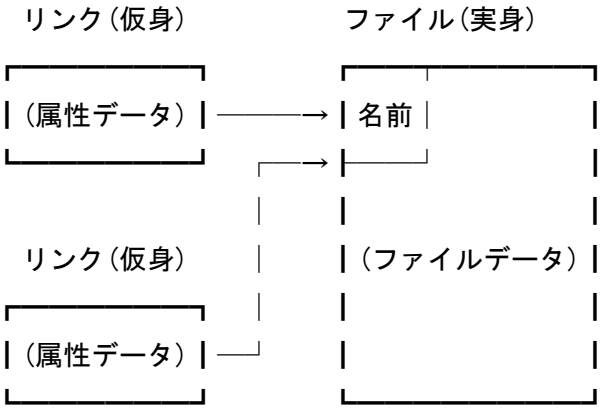
リンクはファイルを参照するための手掛かりとなる一種のポインタであり、参照するファイルを示すデータとリンク独自のいくつかの属性データをひとまとめにしたデータ構造である。

リンクは、1 つのレコードとして任意のファイルの中に埋め込まれて存在する。1 つのファイルを示すリンクは複数個存在しても良く、これにより全体としてネットワーク状の任意のファイル間の参照関係が定義される。

実身／仮身モデルとの対応において、ファイルは実身に、リンクは仮身に 1 対 1 に対応することになる。

ファイルの参照は基本的にリンクを通して直接行われるため、ファイルの名前は絶対的な意味を持たず、1 つの検索キーとして使用される。ファイル名としては最大 20 文字の任意のファイル名が付けられるが、同一のファイル名が存在しても構わない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.



[図 8] ファイルとリンク

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.7.3 ファイルシステム

ファイルシステムはファイルを管理するための 1 つの物理的な単位であり、1 つの記憶媒体上に構築され、物理的なサイズの上限を持つ。

ファイルシステムには、必ず 1 つのルートファイルが存在し、そのルートファイルに含まれるリンクを順次たどっていくことにより、そのファイルシステム内のすべてのファイルに基本的に到達可能である。

ルートファイルは、実身／仮身モデルとの対応においてデバイス実身に対応する。

ファイル間のリンクによる参照関係は、基本的に 1 つのファイルシステム内で定義され整合が取られるが、他のファイルシステムに存在するファイルを参照するための間接的なリンクを持つことができ、これを特に間接リンクと呼ぶ。間接リンクによる参照の場合は、参照先の他のファイルシステムの変更に対する整合は取られないため、参照先のファイルの存在は保証されない場合があるため注意が必要である。

ファイルシステムの生成時には、ファイルシステム名、およびデバイス所在名が設定される。

ファイルシステム名はルートファイルの名称としても設定される最大 20 文字の名前であり、システムおよびユーザがファイルシステムを絶対的に識別するために使用される。ファイルシステム名が同一のファイルシステムは、同一とみなされるため、ファイルシステム名はユニークでなければいけない。

デバイス所在名は、ファイルシステムが格納されている物理デバイスを示す最大 20 文字の名前であり、ネットワーク経由で他のマシンをアクセスする場合や、フロッピー等の装着を求める場合に使用される。

4.7.4 ファイルシステムの接続

システムのスタートアップ時には、ファイルシステムは 1 つも存在していない状態であり、接続操作を行うことにより初めて、ファイルシステムとして使用することが可能となる。従って、通常はシステムの初期化処理として最低限のファイルシステムの接続を行う必要がある。

ファイルシステムの接続は、接続するファイルシステムが存在する論理デバイス名と、接続名を指定して行われる。接続名は接続したファイルシステムを識別するための最大 8 文字の名前であり、接続したファイルシステムのルートファイルを示す絶対パス名称として使用される。また、接続時には接続したファイルシステムのルートファイルへのリンクが得られる。

従って、接続時に得られたリンク、または接続名を使用することにより、接続したファイルシステム上のルートファイルをアクセスすることが可能となり、ルートファイルから順次リンクをたどることにより、接続したファイルシステム上の任意のファイルをアクセスすることが可能となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステムの切断は、切断するファイルシステムの論理デバイス名を指定して行われ、これにより切断したファイルシステム上のファイルを示すリンクを通したファイルのアクセスはできない状態となり、この状態を切断状態と呼ぶ。
実身／仮身モデルとの対応において、切断状態のリンクは虚身に対応することになる。

ファイルシステムの接続は、単にファイルシステムの存在をファイル管理機能に動的に登録するだけの機能であり、構造を持たない平坦な接続である。従って、複数のファイルシステムをまたがるネットワーク状の静的なファイル参照構造は、異なるファイルシステム上のファイルを参照する間接リンクを利用して構築することになる。

4.7.5 ファイル ID

1 つのファイルシステム内のすべてのファイルには､生成時にファイル ID と呼ばれるユニークな番号が付けられ、内部的に識別される。ファイル ID は 0 ～ (最大ファイル ID)の範囲の値であり、最大ファイル ID(即ち、最大のファイル数)はファイルシステムの生成時に規定される｡ファイル ID は 16 ビットの数値で表されるため、最大ファイル ID は 65535 を越えることはできない。

ファイルシステムのルートファイルは常に 0 のファイル ID を持つ。

4.7.6 リンク

リンクは、ファイルをアクセスするための手掛かりとなる一種のポインタであり、参照するファイルが存在するファイルシステム名、ファイル ID 、およびリンクとしてのいくつかの属性データを保持しているデータ構造である。
リンクは単なるポインタとしての動的なデータであるが､ファイル内に 1 つのレコードとして格納することにより固定的な存在となる｡このように格納されたリンクを特に固定 (フィックスド)リンクと呼ぶ｡固定リンクはファイルシステム名を持たないため、同一のファイルシステム内のファイルの参照のみ可能であり、ファイルから固定リンクを取り出した時点で、そのファイルの属するファイルシステム名がリンクのデータ構造として設定されることになる。

このため、異なるファイルシステムを参照するリンクを固定リンクとしてファイル内に格納する場合は、あらかじめ、格納するファイルシステム内にリンクファイルと呼ばれる特殊なファイルを生成し、そのリンクファイルを示すリンクを固定リンクとして格納する必要がある。

リンクファイルは、参照するファイルの存在するファイルシステム名、ファイル ID、ファイル名、および生成日時 を保持している特殊なファイルであり、リンクファイルへのアクセスは、参照している異なるファイルシステム内のファイルへのアクセスと自動的に解釈される。リンクファイルを示すリンクを特に間接リンクと呼び、通常のファイルを示すリンクを直接リンクと呼ぶ。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

多重の間接リンク、即ち 2 つ以上のリンクファイルを経由したファイルの参照はサポートされないため、アクセスした時点でファイルは存在しないというエラーになる。

間接リンクによるリンクファイル経由によるファイルの参照は、以下のように行われる。

・ファイルシステム名によりファイルシステムを特定する。接続されていない場合は、アクセスは不可となる。なお、接続名は、リンクファイル経由のアクセスには無関係である。

・ファイル ID により特定したファイルシステム内のファイルをチェックし、ファイル名および生成日時の両方またはどちらか一方が一致しており、かつそのファイルがリンクファイルでない場合に、そのファイルを対象としてアクセスを行う。それ以外の場合は、参照すべきファイルが存在しなくなったものとみなしてアクセスは不可となる。

4.7.7 作業ファイル

あるプロセスが現在処理の対象としているファイルをそのプロセスの作業ファイルと呼ぶ。プロセスはシステムコールにより任意のファイルを作業ファイルとすることが可能である。

作業ファイルはプロセスの実行環境として保持されており、生成した子プロセスに継承される。

作業ファイルは未定義とすることも可能であり、システムで最初に生成されたプロセスの作業ファイルは未定義状態となっている。

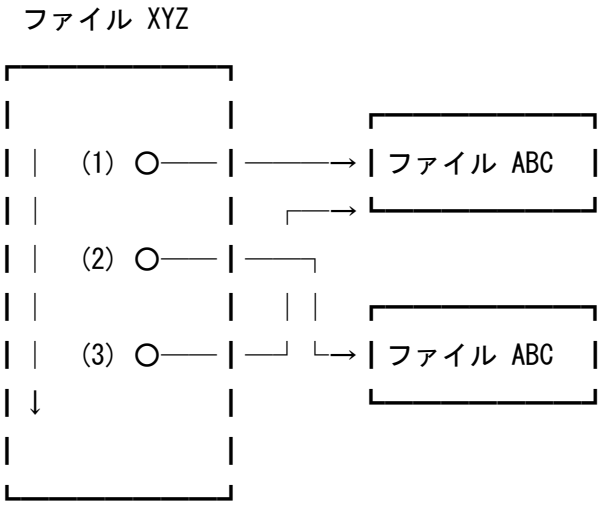
4.7.8 パス名

ファイルの参照は基本的にリンクにより直接的に行われるが、バッチ的なアプリケーション等では、リンクをインタラクティブに順番にたどることができないため、直接的にたどるべきリンクの列を指定することによりファイルを参照することになる。

このためのリンクの列として、各リンクの参照するファイル名を順番に並べたものを、パス名と呼ぶ。この場合、ファイル名だけでは一意性が保証されないため、その出現順をファイル名に付加して使用する。

出現順は、 1 つのファイル内に同一のファイル名を持つファイルを参照するリンクが n 個存在する場合に先頭から付けられた 0 ～ n-1 の連続番号である。出現順を省略した場合は 0、即ち最初とみなされる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.



[図 9] パス名における出現順の例

パス名は以下に示す構文を持ち、最大 256 文字までの 1 つの文字列として取り扱われる。

[パス名]　　::= [特殊参照]|[特殊参照] / [単純パス名]|[単純パス名]
[単純パス名]::= [単純パス名] / [参照指定]| [参照指定]
[参照指定]　::= [ファイル名] |[ファイル名]:[出現順]
[特殊参照]　::= / [接続名]| ≡
[出現順]　　::= 数値
[ファイル名]::= 文字列（最大 20 文字）
[接続名]　　::= 文字列（最大 8 文字）

特殊参照は以下の意味を持つ。

/ [接続名] -- 指定した接続名で接続したファイルシステムのルートファイルを示す。
≡　　-- 作業ファイルを示す。

/ ≡ : の記号は、それぞれ以下に示す特殊コードであるため、 ファイル名としては空白を含めた表示可能なすべての文字が使用可能である。パス名の文字列の最後に / が存在していた場合は、それは無視されるものとする。

/　　TC_FDLM 0xff21
:　　TC_FSEP 0xff22

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

≡ TC_FOWN 0xff23

/ [接続名]で始まるパス名は、そのファイルシステムのルートファイルからのパス名であり、絶対パス名と呼ぶ。それ以外の場合は現在の作業ファイルからの相対的なパス名であり、相対パス名と呼ぶ。

パス名としては以下のようなものがあげられる。

/最新/プロジェクト/ソフトウェア仕様/核仕様/ファイル管理

外部仕様/第 10 章/例:1

≡

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.7.9 ファイルのタイプ

ファイルには、大きく以下の 2 種類のタイプが存在し、単にファイルと言った場合は、通常ファイルを意味する。

通常ファイル　：データの保存場所としての通常の意味でのファイルである。

リンクファイル：別のファイルシステム内のファイルを間接的に参照するために使用される特殊なファイルであり、このファイルを示すリンクは間接リンクである。

4.7.10 通常ファイルの構成

通常ファイルは、任意バイト長のレコードの順序付けられた列、即ちレコードストリームにより構成され、各レコードは、以下の要素により構成される。

・レコードタイプ
・レコードサブタイプ
・レコードサイズ
・レコード本体

レコードタイプは、レコードのタイプを示す 0 ～ 31 の値である。

0　　リンクレコード
　　他のファイルへのリンクを格納するレコードであり、その内容はファイル管理機能により直接的に取り扱われ、アプリケーションからの直接的な操作は限定される。

1～31 データレコード
　　システムとして規定されるレコードタイプであるが、その内容に関してはファイル管理機能としては特に関知せず、単なるバイト列として取り扱う。

レコードサブタイプは、レコードタイプに応じて使用される補助的なタイプ指定や、キーワードに使用される 16 ビットの符号無し数値である。
レコードサイズは、レコード本体のバイト数を示す 32 ビットのデータである。リンクレコードの場合はレコードサイズ情報を持たないが、レコードの入出力に必要な領域のサイズである LINK 構造体のサイズ
（52 バイト）をレコードサイズとする。ただし、このリンクレコードのサイズは、ファイル管理情報としての総バイト数には、カウントされない。
レコード本体は、レコードサイズで示されたバイト数のデータ列であり、その内容はレコードタイプに依存して決められている。リンクレコードの場合はレコード本体は特殊な取り扱いとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

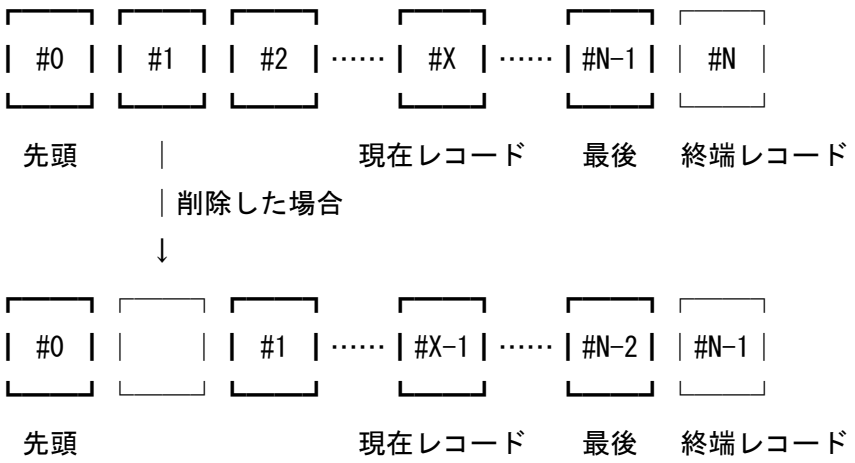
4.7.11 レコード番号／現在レコード

ファイルの各レコードには、先頭のレコードを 0 としたレコードの順番に従った連続番号が付けられており、これをレコード番号と呼ぶ。レコード番号はレコードの順番を示しているためレコードの挿入／削除により動的に変化する。

最後のレコードの次に仮想的にレコードが存在しているものと考え、このレコードを終端レコードと呼ぶ。N 個のレコードがある場合、終端レコードはレコード番号 N を持つことになる。

オープンしたファイルに対しては現在、アクセスの対象としているレコードを示す現在レコードが定義され、現在レコードのレコードに対してデータアクセスが行われる。現在レコードはレコード番号の指定、レコードタイプによる検索等により移動させることができる。

現在レコードはレコードの挿入／削除によっても変化せず、現在レコードに対応するレコード番号が変化することになる。



[図 10] レコード削除時のレコード番号の変化

4.7.12 リンクファイルの構成

リンクファイルは、異なるファイルシステム内に存在するファイルを間接的に参照するために生成され使用されるファイルであり、アプリケーションデータは存在せず、以下に示す管理用データのみが保存されている。

- 参照するファイルのファイル ID
- 参照するファイルのアプリケーションタイプ
- 参照するファイルのファイル名
- 参照するファイルの生成日時
- 参照するファイルが存在したファイルシステム名
- 参照するファイルが存在したファイルシステムのデバイス所在名

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.7.13 ファイル操作

ファイルはプロセスによりアクセスされるが、オープンしたファイルに対しては各プロセス毎で定義されるファイルディスクリプタ(fd > 0) という正の整数値が割り当てられ、そのファイルディスクリプタを使用して実際のファイルアクセスを行う。

プロセスの終了時には、オープンしたファイルはすべて自動的にクローズされる。また、オープンしたファイルに対しては、現在対象としているレコードを示す現在レコードが定義される。

ファイルディスクリプタ、現在レコード位置は、そのプロセス固有のものとして定義され、子プロセスには特に継承されない。

プロセスの環境として作業ファイルは子プロセスに継承される。

4.7.14 ファイルの参照カウント

ファイルには、そのファイルを参照している同一ファイルシステム内の固定リンクの数を示す参照カウントが存在する。参照カウントはファイルを生成した時点では 0 であり、ファイルに対する固定リンクを生成した時点、即ち、リンクをファイル内に格納した時点で +1 される。逆に固定リンクが削除された時点で参照カウントは -1 される。

参照カウントは、同一ファイルシステム内での参照を示すため、リンクファイルを経由したファイルの参照は、参照カウントに反映されないことになる。なお、リンクファイル自体にも参照カウントは適用される。

ファイルの削除は参照カウント 0 のファイルに対してのみ可能である。削除したファイルに固定リンクが含まれていた場合は、その固定リンクが参照しているファイルの参照カウントが -1 されるが、その結果 0 となった場合でも、そのファイルは削除されない。なお、固定リンクが含まれているファイルの削除は、削除時に強制削除の指定を行った時のみ可能となる。

リンクファイルの削除も同様であり、リンクファイル自体の参照カウントが 0 の場合に削除可能となる。なお、リンクファイルの参照先のファイルは、リンクファイル経由で削除することはできない。

ファイルシステムのルートファイルは例外的に参照カウントが最初から 1 となっており、決して削除することができないようになっている。

参照カウント 0 のファイルは、そのファイルを参照する固定リンクを持たないため、動的なリンクが失われてしまうと、通常の方法ではアクセスできないことになるが、ファイルシステム内のすべてのファイルに対するリンクを取り出す方法によりアクセスすることは可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.7.15 ファイルのアクセス

ファイルは、読み込み(READ)／書き込み(WRITE)／更新(UPDATE) のいずれかを指定してオープンするが、オープン時に他からの同一ファイルの同時オープンを制限するための以下のモード指定が可能である。デフォルトは共有モードとなるが、通常は排他書込モードとすることが安全である。

排他モード ： 他からのいかなる同時オープンを一切禁止するモード。
排他書込モード ： 他からの書き込み／更新の同時オープンを一切禁止するモード。
共有モード ： 他からのいかなる同時オープンも禁止しないモード。

以下に既にオープンされているモードに対して新規に同時オープンできるモードの組み合わせを示す。新規の同時オープンが不可の場合はオープン時にエラーとなる。

[表 1] 同時オープン可能なモードの組み合わせ

		新規同時オープンモード								
		排他モード			排他書込モード			共有モード		
既オープンモード		R	W	U	R	W	U	R	W	U
排他モード	R	×	×	×	×	×	×	×	×	×
	W	×	×	×	×	×	×	×	×	×
	U	×	×	×	×	×	×	×	×	×
排他書込モード	R	×	×	×	○	×	×	○	×	×
	W	×	×	×	×	×	×	○	×	×
	U	×	×	×	×	×	×	○	×	×
共有モード	R	×	×	×	○	○	○	○	○	○
	W	×	×	×	×	×	×	○	○	○
	U	×	×	×	×	×	×	○	○	○

また、オープンしたファイルのレコード単位で他からのアクセスを禁止するためのレコードロック機能が用意されている。

ロックしたレコードに対する他からのアクセスは以下のようになる。

・レコードの読み込み、書き込み、置換、サイズ縮小、および削除はエラーとなる。
・サーチの対象、現在レコードとすることは可能。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

既にロックされているレコードをロックしようとした場合はエラーとなるか、またはロックが解かれるまで待たされることになる。

4.7.16 ファイルシステムの管理情報

個々のファイルシステムに対して、以下の内容の管理情報を読み出すことができる。

```
typedef struct {
        UH      fs_bsize;           /* 論理ブロックのバイト数 */
        UH      fs_nfile;           /* 最大ファイル数 */
        H       fs_lang;            /* ファイルシステムでの使用言語 */
        H       fs_level;           /* ファイルシステムのアクセス管理レベル */
        W       fs_nblk;            /* 全体のブロック数 */
        W       fs_nfree;           /* 未使用ブロックの総数 */
        STIME   fs_mtime;           /* 最新のシステムブロックの更新日時 */
        STIME   fs_ctime;           /* ファイルシステムの生成日時 */
        TC      fs_name[L_FSNM];    /* ファイルシステム名 */
        TC      fs_locat[L_DLNM];   /* デバイス所在名 */
} FS_STATE;
```

・fs_bsize は 1 論理ブロックのバイト数であり、2 のべき乗の値とする。

・fs_nfile はそのファイルシステムに登録可能な最大のファイル数を示す。この値は、最大のファイル ID + 1 に等しい。

・fs_lang はファイルシステムでの使用言語を示し、このファイルシステムで使用している文字コード体系を表している。

・fs_level は、ファイルシステムのアクセス管理のレベルを表すもので、以下の値をとる。

0: レベル 0 -- アクセス管理なし
1: レベル 1 -- アクセス管理あり(隠し名なし)
2: レベル 2 -- アクセス管理あり(隠し名あり)

・fs_nblk はファイルシステム全体の論理ブロックの総数であり、この値は論理ブロック番号の最大値 + 1 に等しい。

・fs_nfree は、現時点の未使用論理ブロックの総数であり、このデータは動的に変動する。

・fs_mtime, fs_ctime はそれぞれファイルシステムの最新更新日時と生成された日時で、基準日(1985 年 1 月 1 日 00:00:00 GMT) からの秒数で表す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

・fs_name, fs_locat は、それぞれファイルシステムの生成時に設定される名称であり 20 文字に満たない場合は 0 が詰められている。

ファイルシステムの管理情報はファイルシステムの生成時(フォーマット時) に設定され、以後は未使用ブロックの総数 (fs_nfree) と、最新のシステムブロックの更新日時 (fs_mtime)、ファイルシステム名、およびデバイス所在名を除いて変更されることはない。

4.7.17 ファイルの管理情報

個々のファイルに対して以下の内容の管理情報を読み出すことができる。但し、リンクファイルの場合はリンクファイルが参照するファイルの管理情報となり、リンクファイル自体の管理情報は読み出すことはできない。

ファイル名 :

20 文字のファイル名であり変更可能である。

参照カウント :

ファイルを参照している同一ファイルシステム内の固定リンクの数である。

ファイル管理情報 :

以下に示す各種の管理情報である。

```
typedef struct {
        UH      f_type;         /* ファイルタイプ／所有者アクセスモード */
        UH      f_atype;        /* アプリケーションタイプ */
        TC      f_owner[L_USRNM]; /* ファイル所有者名 (隠し名は常に 0) */
        TC      f_group[L_USRNM]; /* 所有グループ名      (隠し名は常に 0) */
        UH      f_grpacc;       /* グループアクセスレベル */
        UH      f_pubacc;       /* 一般アクセスレベル */
        H       f_nlink;        /* 含まれているリンク数 */
        H       f_index;        /* インデックスレベル */
        W       f_size;         /* ファイルの総バイト数 */
        W       f_nblk;         /* 総使用論理ブロック数 */
        W       f_nrec;         /* 総レコード数 */
        STIME   f_ltime;        /* ファイルの保存期限(日時) */
        STIME   f_atime;        /* 最新のアクセス日時 */
        STIME   f_mtime;        /* 最新の更新日時 */
        STIME   f_ctime;        /* ファイルの作成日時 */
} F_STATE;
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

・f_type はファイルのタイプ、アクセス属性、所有者アクセスモードを示す以下のものである。

TTTT xxxx BAPO xRWE

T: ファイルタイプ

0　　リンクファイル

1　　通常ファイル

2～　　予約

P: 削除不可属性

1 の場合、このファイルが削除禁止であることを示す。

O: 書込不可属性

1 の場合、このファイルが書き込み禁止であることを示す。

A: アプリケーション属性 1

B: アプリケーション属性 2

アプリケーションで規定して使用する属性であり、ファイル管理では、その意味は関知しない。

RWE:　ファイル所有者のアクセスモード(それぞれ、1 の時可)

x :　予約 (0 とする)

・アプリケーションタイプ (f_atype)はアプリケーションが設定／使用するデータであり、ファイル管理では使用しない。

・ 所有者名 (f_owner)、所有グループ名 (f_group)はそれぞれ 12 文字であり、12 文字以下の場合は 0 で埋められる。続く 2 文字の隠し名は常に 0 として得られる。

・グループアクセスレベル(f_grpacc)、一般アクセスレベル(f_pubacc)は以下の構成となる。

xxxx RRRR WWWW EEEE

RRRR : 読み込み可能な最低のユーザレベル (0～15)

WWWW : 書き込み可能な最低のユーザレベル (0～15)

EEEE : 実行可能な最低のユーザレベル　(0～15)

xxxx : 未使用 (0)

・含んでいるリンク数は (f_nlink)はそのファイルが含んでいるリンクレコードの数を示す。

・インデックスレベルは 0 ～ のレコードインデックスの間接の多重度を示す。

・ファイルの総バイト数 (f_size)はファイル内に実際に書かれているデータの総バイト数であり、各レコードのレコードサイズの合計となる。この場合、リンクレコードのレコードサイズは 0 としてカウントされる。

・総使用論理ブロック数は、そのファイルで使用している論理ブロックの総数を示す。

・総レコード数は、そのファイル内に存在するレコードの総数を示す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

・日時に関しては、1985 年 1 月 1 日 0：00 GMT からの秒数が設定される。この値が -1 の場合にはそのデータは無効であることを示す。

最新のアクセス日時（ f_atime）

ファイルのデータを最後にリードした、またはインデックス部を最後に更新した日時。ファイルの生成時には -1（サポートされていない場合）、あるいはファイルの生成日時が設定される。

最新の更新日時（ f_mtime ）

ファイルのデータを最後に更新した日時。ファイルの生成時には生成日時が設定される。

ファイルの作成日時（ f_ctime ）

ファイルを最初に生成した日時。

保存期限（ f_ltime ）

ファイルの保存期限。ファイルの生成時には -1 が設定される。このデータはアプリケーションで設定／使用するもので、ファイル管理では使用しない。

ファイルの所在情報 ：

各ファイルが属するファイルシステムの情報であり、この内容はファイルシステムの管理情報の一部である。

```
typedef struct {
        STIME   fs_ctime;               /* ファイルシステムの生成日時 */
        TC      fs_name[L_FSNM];        /* ファイルシステム名 */
        TC      fs_locat[L_DLNM];       /* デバイス所在名 */
        TC      fs_dev[L_DEVNM];        /* 論理デバイス名 */
} F_LOCATE;
```

・論理デバイス名は、 その時点でファイルシステムが存在しているブロック型デバイスの名称である。

リンクファイル情報 ：

リンクファイルに対してはリンクファイル自体に保持されている以下の参照先のファイルの情報が得られる。この情報は参照先のファイルシステムが接続されていない場合でも取り出すことが可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.7.18 リンクの構造

ファイルをアクセスするために使用される、リンクは以下のデータ構造となる。

```
typedef struct {
        TC      fs_name[L_FSNM]; /* ファイルシステム名 */
        UH      f_id;   /* ファイル ID */
        UH      atr1;   /* 属性データ 1 */
        UH      atr2;   /* 属性データ 2 */
        UH      atr3;   /* 属性データ 3 */
        UH      atr4;   /* 属性データ 4 */
        UH      atr5;   /* 属性データ 5 */
} LINK;
```

・ファイルシステム名は接続されたファイルシステム名そのものであり、 ファイルシステムを絶対的に識別するために使用される。固定リンクとした場合は、この情報はファイルに格納されない。

・ファイル ID はファイルシステム名により特定されたファイルシステム内のファイル ID である。

・属性データ 1～5 はリンク自体として保持される属性データであり、ファイル管理としてはその内容に基本的に関知せず、上位レベルでその用途が決められているものである。新規にリンクを生成した場合のデフォルト値はすべて 0 とされる。このデータは、固定リンクとした場合に、ファイルに格納され、固定リンクを読み込みた場合に、ファイルに格納されている内容が取り出される。

ファイル管理ではファイルシステム名とファイル ID のみを使用して実際のファイルアクセスが行われる。
通常、リンクはファイル管理機能から得られたものを使用するが、アプリケーションがファイルシステム名とファイル ID を直接設定してリンクを作成することも可能である。
例えばファイルシステムのルートファイルへのリンクは、ファイル ID = 0 であるため、ファイルシステム名が判っていればアプリケーションで直接リンクを作成することが可能となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.7.19 システムコール

ファイルのリンク獲得

tkse_get_lnk

C言語インタフェース

```
ER ercd = tkse_get_lnk(TC *path, LINK *lnk, W mode);
```

パラメータ

TC	*path	対象パス名
		NULL　作業ファイルを対象
LINK	*lnk	獲得したリンクの格納領域（出力）
		作業ファイル指定（入力：F_BASED 指定時）
W	mode	リンク獲得モード
		(F_NORM ・ F_BASED) | [F_DIRECT]
		F_NORM　通常指定
		F_BASED　作業ファイル指定
		F_DIRECT　直接リンク獲得指定

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		＝ 0　正常終了(通常ファイルのリンク)
		＝ 1　正常終了(リンクファイルのリンク：F_DIRECT 指定なし)
		＝ 2　正常終了(リンクファイルが参照する通常ファイルのリンク:F_DIRECT 指定時)

エラーコード

E_FACV	パス名(path)内の経路ファイルのアクセス権(E)がない。
E_MACV	アドレス(path,lnk)のアクセスは許されていない。
E_FNAME	パス名(path)が空、不正、または長すぎる。
E_IO	入出力エラーが発生した。
E_NOFS	パス名(path)内のファイル、リンクファイルの参照ファイル(F_DIRECT 指定時)の属するファイルシステムは接続されていない。
E_NOEXS	パス名(path)内のファイル、リンクファイルの参照ファイル(F_DIRECT 指定時)は存在していない、または作業ファイルが未定義。
E_PAR	パラメータが不正である(mode が不正)。
E_SYSMEM	システムのメモリ領域が不足した。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

解説

パス名で指定したファイルのリンクを獲得する。

パス名の指定が NULL の時は現在の作業ファイルのリンクを獲得する。

パス名が相対パス名のとき、F_NORM 指定のときは現在の作業ファイルをベースとするが、F_BASED 指定のときは lnk で指定したファイルを作業ファイルとみなしてベースとする。

指定したファイルがリンクファイルのとき、F_DIRECT 指定なしのときはリンクファイル自体へのリンクを獲得する。このとき、得られたリンクファイルが参照する通常ファイルの存在は保証されない。

F_DIRECT 指定のときはリンクファイルが参照する通常ファイルへの直接のリンクを獲得する。

ファイルのリンクを取り出すためには、パス名に含まれる各ファイルに対しての実行／サーチ(E)アクセス権が必要となるが、対象ファイル自体の実行／サーチ(E)アクセス権は必要ない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

作業ファイル変更

tkse_chg_wrk

C言語インタフェース

```
ER ercd = tkse_chg_wrk(LINK *lnk);
```

パラメータ

LINK	*lnk	変更する作業ファイル
		NULL　作業ファイルを未定義とする

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	ファイル(lnk)のアクセス権(E)がない。
E_MACV	アドレス(lnk)のアクセスは許されていない。
E_IO	入出力エラーが発生した。
E_NOEXS	ファイル(lnk)は存在していない。
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない。
E_SYSMEM	システムのメモリ領域が不足した。

解説

指定したファイルを自プロセスの作業ファイルとする。
作業ファイルとするためには、そのファイルに対しての実行／サーチ(E)アクセス権が必要となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル生成

tkse_cre_fil

C言語インタフェース

```
ER ercd = tkse_cre_fil(LINK *lnk, TC *name, A_MODE *mode, UH atype, W opt);
```

パラメータ

LINK	*lnk	生成したファイルのリンク格納領域(出力)
		ファイルシステム指定（入力：F_FLOAT 指定時）
		親ファイル指定（入力：F_FIX 指定時）
		生成ファイル指定（入力：F_FILEID 指定時）
TC	*name	ファイル名(0 または最大ファイル名文字数まで有効)
A_MODE	*mode	アクセスモード
		NULL　デフォルトアクセスモード適用
UH	atype	ファイルアプリケーションタイプ
W	opt	生成属性
		(FLOAT ・ F_FIX ・ F_FILEID)
		F_FLOAT　浮動リンク指定
		F_FIX　固定リンク指定
		F_FILEID　ファイル ID 指定

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		＞ 0　正常終了(ファイルディスクプリタ)

エラーコード

E_OK	正常終了
E_FACV	ファイル(lnk)のアクセス権(W)がない(F_FIX 指定時)。
E_MACV	アドレス(lnk,name,mode)のアクセスは許されていない。
E_BUSY	ファイル(lnk)は既に排他的にオープンされている為、同時にファイルをオープンすることができなかった(F_FIX 指定時)。
E_OBJ	ファイル(lnk)は既に存在している(F_FILEID 指定時)。
E_FNAME	ファイル名(name)は空または不正である。
E_IO	入出力エラーが発生した。
E_LIMIT	最大ファイル数を越えた、または同時オープン可能な最大ファイル数を越えた。
	ファイル(lnk)のサイズがシステムの制限を越えた(F_FIX 指定時)。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_NODSK	ディスクの領域が不足した。
E_NOEXS	ファイル(lnk)は存在していない(F_FIX 指定時)。
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない。
E_PAR	パラメータが不正である(f_grpno<0,>4 ,opt が不正)。
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である。
E_SYSMEM	システムのメモリ領域が不足した。

解説

lnk で指定したファイル(リンクファイルのときはリンクファイルが参照する通常ファイル)が存在するファイルシステム上に新規に通常ファイルを生成し、更新用にオープンする。

生成したファイルのリンクの属性データはすべて 0 に設定され、lnk で指定した領域に格納される。

F_FLOAT 指定のときは、単純にファイルを生成する。生成したファイルの参照カウントは 0 となる。この場合、lnk で指定した内容のファイルシステム名のみが有効でありファイル ID は無視されるため lnk で指定したファイルは存在していなくてもよい。

F_FIX 指定のときは､生成したファイルのリンクを lnk で指定したファイルの最後のレコード位置にリンクレコード(サブタイプ =0) として追加する。生成したファイルの参照カウントは 1 となる。この場合、lnk で指定したファイルは存在し、かつ書き込みオープンできなくてはいけない。

F_FILEID 指定のときは、lnk で指定した内容のファイル ID と同じファイル ID のファイルを生成する。生成したファイルの参照カウントは 0 となる。この場合、lnk で指定したファイルは存在していてはいけない。

A_MODE は生成したファイルのアクセスモードを指定する。

生成したファイルの所有者アクセスモードが書き込み不可のときでも、ファイルは更新用にオープンされ、そのレコード番号は 0 となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

リンクファイルの生成

tkse_cre_lnk

C言語インタフェース

```
ER ercd = tkse_cre_lnk(LINK *lnk, F_LINK *ref, W opt);
```

パラメータ

LINK	*lnk	生成したファイルのリンク格納領域(出力)
		ファイルシステム指定　(入力：F_FLOAT 指定時)
		親ファイル指定　(入力：F_FIX 指定時)
		生成ファイル指定　(入力：F_FILEID 指定時)
F_LINK	*ref	生成するリンクファイルの内容
W	opt	生成属性
		(F_FLOAT ・ F_FIX ・ F_FILEID)
		F_FLOAT　浮動リンク指定
		F_FIX　固定リンク指定
		F_FILEID　ファイル ID 指定

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	ファイル(lnk)のアクセス権(W)がない(F_FIX 指定時)。
E_MACV	アドレス(lnk,ref)のアクセスは許されていない。
E_BUSY	ファイル(lnk)は既に排他的にオープンされている為、同時にファイルをオープンすることができなかった(F_FIX 指定時)。
E_OBJ	ファイル(lnk)は既に存在している(F_FILEID 指定時)。
E_FNAME	ファイル名(ref->f_name)ファイルシステム名(ref->fs_name)は空または不正である。
E_IO	入出力エラーが発生した。
E_LIMIT	最大ファイル数を越えた。
	ファイル(lnk)のサイズがシステムの制限を越えた(F_FIX 指定時)。
E_NODSK	ディスクの領域が不足した。
E_NOEXS	ファイル(lnk)は存在していない(F_FIX 指定時)。
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない。
E_PAR	パラメータが不正である(opt が不正、同一ファイルシステム)。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_RONLY　　ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である(F_FIX 指定時)。

E_SYSMEM　　システムのメモリ領域が不足した。

解説

lnk で指定したファイル(リンクファイルのときはリンクファイルが参照する通常ファイル)が存在するファイルシステム上に新規に ref で指定した内容のリンクファイルを生成する。

生成したリンクファイルのリンクの属性データはすべて 0 に設定され、lnk で指定した領域に格納される。

F_FLOAT, F_FIX, F_FILEID の意味は tkse_cre_fil() と同一である。

生成したリンクファイルの内容は ref で指定した内容となるが、その生成日時は、ref->f_ctime ではなく、リンクファイルを生成した日時となる。

ref で指定したファイルの実際の存在はチェックされない。

ref->fs_name が lnk で指定したファイルシステム名と同一の場合、リンクファイルは生成できないためエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルの直接生成

tkse_gen_fil

C言語インタフェース

```
ER ercd = tkse_gen_fil(LINK *lnk, TC *name, F_STATE *stat, F_LINK *ref, W opt);
```

パラメータ

LINK	*lnk	生成したファイルのリンク格納領域(出力)
		ファイルシステム指定　(入力：F_FLOAT 指定時)
		親ファイル指定　(入力：F_FIX 指定時)
		生成ファイル指定　(入力：F_FILEID 指定時)
TC	*name	ファイル名(0 または最大ファイル文字数まで有効)
		(通常ファイル生成時のみ有効、このとき name が NULL ならばエラーとなる)
		(リンクファイルの生成の場合は一切参照されない)
F_STATE	*stat	生成するファイルの内容
F_LINK	*ref	生成するリンクファイルの内容
		(リンクファイル生成時のみ有効)
W	opt	生成属性
		(F_FLOAT ・ F_FIX ・ F_FILEID)
		F_FLOAT　浮動リンク指定
		F_FIX　固定リンク指定
		F_FILEID　ファイル ID 指定

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		＝ 0　正常終了(リンクファイル生成時)
		＞ 0　正常終了(ファイルディスクプリタ：通常ファイル生成時)

エラーコード

E_FACV　レベル 0 のユーザでない

E_MACV　アドレス(lnk,ref,name,stat)のアクセスは許されていない

E_BUSY　ファイル(lnk)は既に排他的にオープンされている為、同時にファイルをオープンすることができなかった(F_FIX 指定時)

E_OBJ　ファイル(lnk)は既に存在している(F_FILEID 指定時)

E_FNAME　ファイル名(name)、ファイル名(ref->f_name)、ファイルシステム名(ref->fs_name)は空または不正である

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_IO	入出力エラーが発生した
E_LIMIT	最大ファイル数を越えた、または同時オープン可能な最大ファイル数を越えた ファイル(lnk)のサイズがシステムの制限を越えた(F_FIX 指定時)
E_NODSK	ディスクの領域が不足した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_PAR	パラメータが不正である(opt が不正、同一ファイルシステム、ref, stat の内容が不正)
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である
E_SYSMEM	システムのメモリ領域が不足した

解説

lnk で指定したファイル(リンクファイルのときはリンクファイルが参照する通常ファイル)が存在するファイルシステム上に新規に通常ファイルまたはリンクファイルを生成し、通常ファイルのときは更新用にオープンする。

生成したリンクファイルのリンクの属性データはすべて 0 に設定され、lnk で指定した領域に格納される。

F_FLOAT, F_FIX, F_FILEID の意味は tkse_cre_fil() と同一である。

生成したファイルの内容は stat で指定し､stat->f_type により通常ファイルかリンクファイルか区別される。

通常ファイル生成のときは、name で指定した名前の通常ファイルを生成し、生成したファイルの管理情報を stat で指定した内容に設定する。ただし、f_nlink, f_index, f_size, f_nblk, f_nrec の値は無視されファイルの生成時に初期化される。

リンクファイル生成のときは、stat の他の内容はすべて無視され、ref の内容のリンクファイルを生成する、tkse_cre_lnk() と同様の動作であるが、ref->f_ctime も有効となる。

ファイルの直接生成は、ファイルシステムの復元などの特殊な用途に使用されるため、ユーザレベル 0 のプロセスでのみ実行可能である。

通常ファイルを生成したときはファイルは更新用にオープンされる｡この状態ではレコードは 1 つも存在しないため現在レコードは終端レコードであり、そのレコード番号は 0 となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルのオープン

tkse_opn_fil

C言語インタフェース

```
ER ercd = tkse_opn_fil(LINK *lnk, W o_mode, TC *pwd);
```

パラメータ

LINK	*lnk	対象ファイル
W	o_mode	オープンモード
		(F_READ ・ F_WRITE ・ F_UPDATE) | [F_EXCL ・ F_WEXCL]
		F_READ　読み込み用オープン
		F_WRITE　書き込み用オープン
		F_UPDATE　更新(読み込み／書き込み)用オープン
		F_EXCL　排他モード
		F_WEXCL　排他書き込みモード
TC	*pwd	パスワード
		NULL　パスワード指定なし

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		＞ 0　正常終了(ファイルディスクプリタ)

エラーコード

E_FACV	ファイル(lnk)のアクセス権(o_mode に対応)がない
E_MACV	アドレス(lnk,pwd)のアクセスは許されていない
E_BUSY	ファイル(lnk)は既に排他的にオープンされている為、同時にファイルをオープンすることができなかった
E_IO	入出力エラーが発生した
E_LIMIT	同時オープン可能な最大ファイル数を越えた
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_PAR	パラメータが不正である(o_mode が不正)
E_PWD	ファイル(lnk)の合言葉が不一致である
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である
E_SYSMEM	システムのメモリ領域が不足した

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

解説

lnk で指定したファイルを指定したモードでオープンする。ファイルをオープンするためにはオープンモードに対応するアクセス権が必要である。

パスワードの機能は現在未サポートのため、pwd には NULL を設定する。

オープンしたファイルの先頭レコードが現在レコードとなる。レコードが 1 つも存在しないときは終端レコードが現在レコードとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルのクローズ

tkse_cls_fil

C言語インタフェース

```
ER ercd = tkse_cls_fil(W fd);
```

パラメータ

W	fd	ファイルディスクリプタ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FD	ファイルディスクリプタは存在していない。
E_IO	入出力エラーが発生した。

解説

オープンしたファイルをクローズする。
ファイルをオープンしたプロセスが終了したときには、自動的にそのファイルはクローズされる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルの削除

tkse_del_fil

C言語インタフェース

```
ER ercd = tkse_del_fil(LINK *org, LINK *lnk, W force);
```

パラメータ

LINK	*org	削除対象ファイルの親ファイル NULL　親ファイル指定なし
LINK	*lnk	削除対象ファイル
W	force	強制削除指定 ＝0 : 削除対象ファイルがリンクレコードを含むときは削除しない。 ≠0 : 削除対象ファイルがリンクレコードを含むときも削除する。

リターンパラメータ

ER	ercd	＜ 0　エラーコード ≧ 0　正常終了(削除した結果参照カウント0となったリンクレコードの数)

エラーコード

E_FACV　ファイル(org)のアクセス権(W)がない(org≠NULL の時)

E_MACV　アドレス(org,lnk)のアクセスは許されていない

E_BUSY　ファイル(org)は既に排他的にオープンされている為、同時にファイルをオープンすることができなかった(org≠NULL の時)。

ファイル(lnk)はオープン中である、または作業ファイルである。

ファイル(lnk)の参照カウントは0でない(org=NULL の時)。

ファイル(lnk)を示すリンクレコードは他のオープンで現在レコードとして使用されている(org=NULL の時)。

E_IO　入出力エラーが発生した

E_LOCK　ファイル(lnk)を示すリンクレコードは、他からロックされている(org≠ NULL の時)

E_NOEXS　ファイル(org,lnk)は存在していない(または、org 内に指定されたファイル(lnk)を示すリンクレコードが存在しない)

E_NOFS　ファイル(org,lnk)の属するファイルシステムは接続されていない

E_PERM　ファイル(lnk)は削除不可である(削除不可属性がセットされている)

E_REC　ファイル(lnk)は リンクレコードを含んでいる(force=0 の時)

E_RONLY　ファイル(org)は書込不可である、または属するファイルシステムは書込不可である(org≠NULL の時)

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

	ファイル(lnk)の属するファイルシステムは書込不可である
E_SYSMEM	システムのメモリ領域が不足した

解説

org で指定した親ファイル内の lnk で指定したファイルを示すリンクレコードを削除し､そのファイルの参照カウントを -1 する。その結果、参照カウントが 0 になった場合には、lnk で指定されたファイル自体を削除する。この場合、親ファイルの書き込み(W)アクセス権が必要となる。

親ファイルの指定なし（ org = NULL ）のときは、lnk で指定したファイルの参照カウントが 0 の場合にファイルを削除する。参照カウントが 0 でない場合はエラーとなる。

削除対象ファイルがリンクファイルのときは、リンクファイル自体が削除の対象となり、リンクファイルの参照先のファイルは削除されることはない。

強制削除指定なし（ force = 0)のときは、 削除対象ファイルがリンクレコードを含むときはエラーとして削除しない。強制削除指定あり（ force ≠0)のときは、削除対象ファイルがリンクレコードを含むときも削除し、含まれるリンクレコードが示すファイルの参照カウントを -1 し、その結果、参照カウントが 0 となったリンクレコードの数をリターン値とする。

削除対象ファイルが以下のいずれかのときは削除されずにエラーとなる。

・削除不可属性が設定されているとき
・オープンしているプロセスが存在しているとき
・作業ファイルとしているプロセスが存在しているとき

書込不可属性がセットされていた場合でも削除可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

現在レコード移動

tkse_see_rec

C言語インタフェース

```
ER ercd = tkse_see_rec(W fd, W offset, W mode, W *recnum);
```

パラメータ

W	fd	ファイルディスクリプタ
W	offset	移動オフセット
W	mode	移動モード
		＝ 0　現在レコード番号 + offset のレコード番号位置に移動
		＞ 0　offset のレコード番号位置に移動
		offset≧0 でなくてはいけない
		＜ 0　終端レコード番号 ＋ offset のレコード番号位置に移動
		offset≦0 でなくてはいけない
W	*recnum	移動後の現在レコード番号の格納領域
		NULL　格納しない

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(recnum)のアクセスは許されていない
E_FD	ファイルディスクリプタは存在していない
E_IO	入出力エラーが発生した
E_REC	存在するレコードの範囲を越えた

解説

オープンしたファイルの現在レコード位置を指定した位置に移動する。
指定した移動先が存在するレコードの範囲を越える場合はエラーとなり、現在レコードは変化しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコード検索

tkse_fnd_rec

C言語インタフェース

```
ER ercd = tkse_fnd_rec(W fd, W mode, UW typemask, UH subtype, W *recnum);
```

パラメータ

W	fd	ファイルディスクリプタ	
W	mode	検索モード(検索開始位置／方向の指定)	
		(F_FWD ・ F_NFWD ・ F_BWD ・ F_NBWD ・ F_TOPEND ・ F_ENDTOP)	
		F_FWD	現在レコードから終端レコードまで
		F_NFWD	現在レコードの次のレコードからから終端レコードまで
		F_BWD	現在レコードから先頭レコードまで
		F_NBWD	現在レコードの前のレコードから先頭レコードまで
		F_TOPEND	先頭レコードから終端レコードまで
		F_ENDTOP	終端レコードから先頭レコードまで
UW	typemask	検索対象レコードタイプのビットマスク	
		LSB	タイプ 0 に対応
		MSB	タイプ 31 に対応
UH	subtype	検索対象レコードサブタイプ	
		0	全サブタイプが対象(サブタイプのチェックなし)
W	*recnum	検索結果の現在レコード番号の格納領域	
		NULL	格納しない

リターンパラメータ

ER	ercd	＜ 0	エラーコード
		≧ 0	正常終了(検索したレコードタイプ)

エラーコード

E_MACV	アドレス(recnum)のアクセスは許されていない
E_FD	ファイルディスクリプタは存在していない
E_IO	入出力エラーが発生した
E_PAR	パラメータが不正である(mode が不正)
E_REC	指定した検索条件に合うレコードは存在しない(typemask=0 の場合も含む)

解説

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

オープンしたファイル内の指定したレコードを検索し、見つけたレコードを現在レコードとする。
対象レコードが見つからなかったときはエラーとなり、現在レコードは変化しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

リンクレコード検索

tkse_fnd_lnk

C言語インタフェース

```
ER ercd = tkse_fnd_lnk(W fd, W mode, LINK *lnk, UH subtype, W *recnum);
```

パラメータ

W	fd	ファイルディスクリプタ
W	mode	検索モード(検索開始位置／方向／内容の指定)
		(F_FWD ・ F_NFWD ・ F_BWD ・ F_NBWD ・ F_TOPEND ・ F_ENDTOP)
		| [F_SFILE] | [F_SNAME]
		| [F_SATR1] | [F_SATR2] | [F_SATR3] |
		[F_SATR4] | [F_SATR5]
		F_FWD～F_ENDTOP tkse_fnd_rec()と同じ
		F_SFILE lnk と同一のファイルを示すリンクレコード
		F_SNAME lnk と同一のファイル名のファイルを示すリンクレコード
		F_SATR1 lnk と同一の属性データ 1 を持つリンクレコード
		F_SATR2 lnk と同一の属性データ 2 を持つリンクレコード
		F_SATR3 lnk と同一の属性データ 3 を持つリンクレコード
		F_SATR4 lnk と同一の属性データ 4 を持つリンクレコード
		F_SATR5 lnk と同一の属性データ 5 を持つリンクレコード
LINK	*lnk	検索対象リンク
		F_SFILE～F_SATR5 を指定したときのみ有効
UH	subtype	検索対象レコードサブタイプ
		0 全サブタイプが対象(サブタイプのチェックなし)
W	*recnum	検索結果の現在レコード番号の格納領域
		NULL 格納しない

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(lnk,recnum)のアクセスは許されていない(lnk は検索条件を指定した場合のみアクセスされる)
E_FD	ファイルディスクリプタは存在していない

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_PAR	パラメータが不正である(mode が不正)
E_REC	指定した検索条件に合うレコードは存在しない
E_SYSMEM	システムのメモリ領域が不足した

解説

オープンしたファイル内の指定したリンクレコードを検索し、見つけたリンクレコードを現在レコードとする。対象レコードが見つからなかったときはエラーとなり、現在レコードは変化しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコード読み込み

tkse_rea_rec

C言語インタフェース

```
ER ercd = tkse_rea_rec(W fd, W offset, B *buf, W size, W *r_size, UH *subtype);
```

パラメータ

W	fd	ファイルディスクリプタ
W	offset	読み込み開始バイト位置(≧0)
B	*buf	読み込みデータ格納領域
		NULL　格納しない
W	size	読み込みデータ格納領域のバイトサイズ(≧0)
W	*r_size	開始バイト位置からの残りバイトサイズ
		(レコードサイズ － offset)の格納領域
		NULL　格納しない
UH	*subtype	レコードサブタイプの格納領域
		NULL　格納しない

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		≧ 0　正常終了(現在レコードのレコードタイプ)

エラーコード

E_MACV	アドレス(buf,r_size,subtype)のアクセスは許されていない
E_ENDR	現在レコードは終端レコードである
E_FD	ファイルディスクリプタは存在していない、または F_WRITE オープンである。
E_IO	入出力エラーが発生した
E_LOCK	現在レコードは他からロックされている
E_PAR	パラメータが不正である(size<0,offset<0,リンクレコードで offset,size が不正)

解説

オープンしたファイルの現在レコードを読み込む。

レコードサイズ ＜offset＋size のときは、buf には (レコードサイズ － offset)バイトのデータのみ読み込まれて格納される。

offset ≧ レコードサイズ、buf＝NULL、または size＝0 のときは buf には何も格納されずに *r_size、

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

*subtype に対応する値が格納される。これは、レコードサイズやサブタイプのみを取り出す場合に使用される。

現在レコードがリンクレコードのときは、LINK 構造体全体の内容が buf に読み出され、*r_size には LINK 構造体のサイズが格納される。この場合、offset = 0、size ≧ LINK 構造体のサイズ（ または size = 0 ）でなくてはいけない。

現在レコードが終端レコードのとき、または他プロセスからロックされているときはエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコード書き込み

tkse_wri_rec

C言語インタフェース

```
ER ercd = tkse_wri_rec(W fd, W offset, B *buf, W size, W *r_size, UH *subtype, UW units);
```

パラメータ

W	fd	ファイルディスクリプタ
W	offset	書き込み開始バイト位置(−1≦ offset ＜レコードサイズ)
		-1：レコードの最後への書き込み(追加)
B	*buf	書き込みデータへのポインタ
		NULL　書き込まない
W	size	書き込みデータのバイトサイズ(≧0)
W	*r_size	開始バイト位置からの残りバイトサイズ
		(書き込み後のレコードサイズ − offset)の格納領域
		NULL　格納しない
UH	*subtype	変更するレコードサブタイプへのポインタ
		NULL　変更しない
UW	units	ブロック獲得単位(Kバイト)
		0　任意

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(buf,r_size,subtype)のアクセスは許されていない
E_ENDR	現在レコードは終端レコードである
E_FD	ファイルディスクリプタは存在していない、または F_READ オープンである。
E_IO	入出力エラーが発生した
E_LOCK	現在レコードは他からロックされている
E_NODSK	ディスクの領域が不足した、または指定された連続ブロック領域が獲得できなかった。
E_PAR	パラメータが不正である(size<0,offset が不正,リンクレコードで offset,size が不正)。
E_LIMIT	ファイルのサイズがシステムの制限を越えた

解説

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

オープンしたファイルの現在レコードに書き込む。

レコードサイズ < offset + size のときは、書き込み後にレコードサイズは増加する。

units はレコードサイズが増加したために必要となった追加ブロックの獲得の単位をKバイト単位で指定するもので、units サイズ以上 (size 以下)の連続ブロック領域を割り当てることを指定する。

units = 0 はブロックの割り当て方法は任意であることを意味する。

size = 0、または buf = NULL のときは、データの書き込みは行わないが subtype ≠ NULL であればレコードサブタイプは変更する。

buf = NULL で レコードサイズ < offset + size のときは、レコードサイズを増加するが増加した部分のデータは不定となる。これは units 指定と組み合わせてレコードの追加ブロック領域をあらかじめ確保するときに使用する。

offset = -1 のときは、常にその時点のレコードの最後にデータを書き込み、*r_size には size の値が格納される。同一レコードを複数のプロセスがオープンして同時に書き込む場合でもこの指定により別のプロセスが書き込んだデータを上書きしないことが保証される。

現在レコードがリンクレコードのときは buf の内容は LINK 構造体となるが、属性データ部分のみが書き込まれ、参照するファイル自体を変更することはできない。この場合、offset = 0、size ≧ LINK 構造体のサイズ(または size = 0)でなくてはいけない。

現在レコードが終端レコードのとき、または他プロセスからロックされているときはエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコード挿入

tkse_ins_rec

C言語インタフェース

```
ER ercd = tkse_ins_rec(W fd, B *buf, W size, W type, UH subtype, UW units);
```

パラメータ

W	fd	ファイルディスクリプタ
B	*buf	挿入レコードのデータへのポインタ
		NULL　データは書き込まない
W	size	挿入レコードのバイトサイズ(≧0)
W	type	挿入レコードのレコードタイプ
UH	subtype	挿入レコードのレコードサブタイプ
UW	units	ブロック獲得単位(Kバイト)
UH	subtype	挿入レコードのレコードサブタイプ
		0　任意

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(buff)のアクセスは許されていない
E_FD	ファイルディスクリプタは存在していない、または F_READ オープンである。
E_IO	入出力エラーが発生した
E_LIMIT	リンクの参照先のファイルの参照カウントがシステムの制限(255)を越えた
	ファイルのサイズがシステムの制限を越えた
E_NODSK	ディスクの領域が不足した、または指定された連続ブロック領域が獲得できなかった。
E_NOEXS	リンクの参照先のファイルは存在していない
E_PAR	パラメータが不正である(type が不正、size<0、units が不正、type=0 の時の size,buff が不正)
E_REC	リンクは別ファイルシステムを参照している

解説

オープンしたファイルの現在レコードの直前に新規のレコードを挿入する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

units は挿入したレコードで必要とするブロックの獲得の単位を K バイト単位で指定するもので、units サイズ以上 (size 以下)の連続ブロック領域を割り当てることを指定する。units = 0 はブロックの割り当て方法は任意であることを意味する。

buf = NULL のときは、挿入したレコードのサイズは size となるが、そのデータは不定となる。これは units 指定と組み合わせて、レコードのブロック領域をあらかじめ確保するために使用する。

type = 0 のときはリンクレコードの挿入であり、buf の内容は LINK 構造体となる。リンクレコードの挿入により、そのリンクの示すファイルの参照カウントは +1 される。この場合、buf ≠ NULL、size = LINK 構造体のサイズ、かつリンクの示すファイルは同一ファイルシステム上に存在していなくてはいけない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコード追加

tkse_apd_rec

C言語インタフェース

```
ER ercd = tkse_apd_rec(W fd, B *buf, W size, W type, UH subtype, UW units);
```

パラメータ

W	fd	ファイルディスクリプタ
B	*buf	追加レコードのデータへのポインタ
		NULL　データは書き込まない
W	size	追加レコードのバイトサイズ(≧0)
W	type	追加レコードのレコードタイプ
U	subtype	追加レコードのレコードサブタイプ
UW	units	ブロック獲得単位(Kバイト)
		0　任意

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(buff)のアクセスは許されていない
E_FD	ファイルディスクリプタは存在していない、または F_READ オープンである
E_IO	入出力エラーが発生した
E_LIMIT	リンクの参照先のファイルの参照カウントがシステムの制限(255)を越えた
	ファイルのサイズがシステムの制限を越えた
E_NODSK	ディスクの領域が不足した、または指定された連続ブロック領域が獲得できなかった
E_NOEXS	リンクの参照先のファイルは存在していない
E_PAR	パラメータが不正である(type が不正、size<0、units が不正、type=0 の時の size,buff が不正)
E_REC	リンクは別ファイルシステムを参照している

解説

オープンしたファイルの最後に新規のレコードを挿入する。
現在レコードの位置の関係なく常に最後のレコード(終端レコードの直前) にレコードを挿入する点を

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

除いて tkse_ins_rec() と同一である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコード削除

tkse_del_rec

C言語インタフェース

```
ER ercd = tkse_del_rec(W fd);
```

パラメータ

W	fd	ファイルディスクリプタ

リターンパラメータ

ER	ercd	＜ 0	エラーコード
		＝ 1	正常終了(リンクレコードを削除した結果、参照カウント=0 となった)
		＝ 0	正常終了(上記以外)

エラーコード

E_BUSY	他のオープンで現在レコードとして使用されている
E_ENDR	現在レコードは終端レコードである
E_FD	ファイルディスクリプタは存在していない、または F_READ オープンである。
E_IO	入出力エラーが発生した
E_LOCK	現在レコードは他からロックされている
E_SYSMEM	システムのメモリ領域が不足した

解説

オープンしたファイルの現在レコードを削除し、削除した次のレコードに現在レコードをを移動する。

削除したレコードがリンクレコードのときは、リンクレコードが示すファイルの参照カウントは -1 され、その結果参照カウントが 0 となった場合はリターン値は 1 となる。

現在レコードが終端レコードのとき、他プロセスからロックされているとき、または他のオープンで現在レコードとなっているときはエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコードサイズ縮小

tkse_trc_rec

C言語インタフェース

```
ER ercd = tkse_trc_rec(W fd, W size);
```

パラメータ

W	fd	ファイルディスクリプタ
W	size	縮小するレコードバイトサイズ(≧0)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ENDR	現在レコードは終端レコードである
E_FD	ファイルディスクリプタは存在していない、または F_READ オープンである。
E_IO	入出力エラーが発生した
E_LOCK	現在レコードは他からロックされている
E_PAR	パラメータが不正である(size<0)
E_REC	現在レコードはリンクレコードである

解説

オープンしたファイルの現在レコードのレコードサイズを size バイトに縮小する。レコードサイズ≦ size のときは何もしない。

現在レコードが終端レコードのとき、リンクレコードのとき、または他プロセスからロックされているときはエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルの内容交換

tkse_xch_fil

C言語インタフェース

```
ER ercd = tkse_xch_fil(W fd_1, W fd_2);
```

パラメータ

W	fd_1	ファイルディスクリプタ 1
W	fd_2	ファイルディスクリプタ 2

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FD	ファイルディスクリプタは存在していない、または (F_UPDATE|F_EXCL)でない オープンである。
E_IO	入出力エラーが発生した
E_LOCK	現在レコードは他からロックされている(レコードにロックがかかっている)。
E_PAR	パラメータが不正である(fd_1 と fd_2 は同一ファイル)
E_XFS	ファイル(fd_1,fd_2)は異なるファイルシステムに属している
E_SYSMEM	システムのメモリ領域が不足した

解説

オープンした 2 つのファイルの内容を交換する。

交換するのはファイルのデータ部分であり、ファイルの管理情報はアクセス日付と更新日付を除いて元のままとなる。

交換する 2 つのファイルは、同一のファイルシステムに存在し、かつ排他モードで更新用オープンしてなくてはいけない。

交換後の現在レコードは、それぞれ先頭レコードとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコードロック

tkse_loc_rec

C言語インタフェース

```
ER ercd = tkse_loc_rec(W fd, W mode);
```

パラメータ

W	fd	ファイルディスクリプタ
W	mode	ロックモード
		(F_UNLOCK ・ F_LOCK ・ F_TSLOCK ・ F_CKLOCK)
		F_LOCK　ロック設定(待ち)
		F_UNLOCK　ロック解除
		F_TSLOCK　ロック設定(待ちなし)
		F_CKLOCK　ロック状態チェック

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ENDR	現在レコードは終端レコードである
E_FD	ファイルディスクリプタは存在していない
E_IO	入出力エラーが発生した
E_LIMIT	同時にロック可能なレコードの最大数を越えた
E_LOCK	現在レコードは他からロックされている現在レコードは既に他からロックされている(F_TSLOCK／F_CKLOCK 指定時)。 自プロセスの他ファイルディスクリプタから既にロックされている(F_LOCK 指定時)。 他ファイルディスクリプタからのロックであり解除できない(F_UNLOCK 指定時)。
E_DISWAI	メッセージハンドラが起動されたため待ち処理が中断された(F_LOCK 指定時)
E_PAR	パラメータが不正である(mode が不正)
E_SYSMEM	システムのメモリ領域が不足した

解説

オープンしたファイルの現在レコードに対するロック操作を行う。
F_LOCK（ロック設定(待ち））が指定された場合は、他プロセスからロックされていたときはロックが解除

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

されるまで待つ(待ちはプロセス優先度順で同一優先度の場合は待ちに入った順)。自プロセスの同一ファイルディスクリプタからロックしていた場合は何もせずに正常終了。自プロセスの他ファイルディスクリプタからロックしていた場合はエラー。

F_UNLOCK（ロック解除）が指定された場合は、レコードがロックされていない場合は何もせずに正常終了。自プロセスの同一ファイルディスクリプタからロックした場合のみ解除可能で、その他の場合はエラー。

F_TSLOCK（ロック設定(待ちなし））が指定された場合は、他プロセスまたは他ファイルディスクリプタからロックされていたときはエラー。

F_CKLOCK（ロック状態チェック）が指定された場合は、他プロセスまたは他ファイルディスクリプタからロックされていたときはエラー、そうでないときは何もせずに正常終了。

ロックしたレコードは、ロックしたファイルディスクリプタ以外からの読み込み、書き込み、サイズ変更、および削除が禁止される。

ファイルをクローズ時にはオープンしたプロセスで設定したロックは解除される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルアクセス権チェック

tkse_chk_fil

C言語インタフェース

```
ER ercd = tkse_chk_fil(LINK *lnk, W mode, TC *pwd);
```

パラメータ

LINK	*lnk	対象ファイル
W	mode	チェックモード ([F_READ] | [F_WRITE] | [F_EXCUTE]) ・ [F_EXIST]
		F_READ　読み込み(R)アクセス権チェック
		F_WRITE　書き込み(W)アクセス権チェック
		F_EXCUTE　実行／サーチ(E)アクセス権チェック
		F_EXIST　ファイルの存在チェック
TC	*pwd	パスワード(F_READ または F_WRITE 指定時のみ有効)
		NULL　パスワード指定なし

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		＝ 0　正常終了(F_EXIST 以外指定時)
		≧ 0　正常終了(ファイルのアクセス情報：F_EXIST 指定時)

エラーコード

E_FACV	ファイル(lnk)のアクセス権(F_EXIST 以外指定時)がない
E_MACV	アドレス(lnk)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_PAR	パラメータが不正である(mode が不正)
E_PWD	ファイル(lnk)の合言葉が不一致である(F_READ/F_WRITE 指定時)
E_RONLY	ファイル(F_WRITE 指定時)の書込不可属性が設定されている、または属するファイルシステムは書込不可である。
E_SYSMEM	システムのメモリ領域が不足した

解説

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

指定したファイルの指定したアクセスが可能か否かのチェックを行う。
F_READ、F_WRITE、F_EXCUTE の組み合わせで指定したアクセスが不可のときはエラーとなる。
パスワードの機能は現在未サポートのため、pwd には NULL を設定する。

F_EXIST 指定のときは、ファイルが存在しない場合はエラーとなり、存在する場合は以下のアクセス情報をリターン値として戻す。

```
0.....0 BAPO SRWE
```

B: アプリケーション属性 2　　(1:ON、0:OFF)
A: アプリケーション属性 1　　(1:ON、0:OFF)
P: 削除不可属性　　(1:ON、0:OFF)
O: 書込不可属性　　(1:ON、0:OFF)
S: パスワードの有無　　(1:有、0:無)
R: 読み込み(R)アクセス権　　(1:有、0:無)
W: 書き込み(W)アクセス権　　(1:有、0:無)
E: 実行／サーチ(E)アクセス権　　(1:有、0:無)

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルアクセスモード変更

tkse_chg_fmd

C言語インタフェース

```
ER ercd = tkse_chg_fmd(LINK *lnk, A_MODE *mode);
```

パラメータ

LINK	*lnk	対象ファイル
A_MODE	*mode	変更するアクセスモード

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	ファイル(lnk)の所有者でない、またはレベル0のユーザでない
E_MACV	アドレス(lnk,mode)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_PAR	パラメータが不正である(mode の内容が不正)
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である。
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したファイルのアクセスモードを変更する。

アクセスモードの以下のそれぞれのデータに対して F_NOCHG 指定を行うと、その項目は変更しないことを意味する。

・所有者アクセスモード(f_ownac)
・グループアクセスレベル(f_grpacc)
・一般アクセスレベル(f_pubacc)
・所有グループ番号(f_grpno)

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

アクセスモードの変更は、ファイルシステムのアクセスレベルが 0 のときは誰でも変更可能であるが、アクセスレベルが 0 でないときはファイルの所有者のプロセスのみ変更可能となる。

すでにオープンされているファイルのアクセスモードを変更した場合、その変更はすでにオープンされているものには影響を与えない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルのアクセス属性変更

tkse_chg_fat

C言語インタフェース

```
ER ercd = tkse_chg_fat(LINK *lnk, W attr);
```

パラメータ

LINK	*lnk	対象ファイル
W	attr	変更するアクセス属性 (F_SETRONLY ・ F_RSTRONLY ・ F_SETPERM ・ F_RSTPERM ・ F_SETA1 ・ F_RSTA1 ・ F_SETA2 ・ F_RSTA2) F_SETRONLY　書き込み不可属性のセット F_RSTRONLY　書き込み不可属性のリセット F_SETPERM　削除不可属性のセット F_RSTPERM　削除不可属性のリセット F_SETA1　アプリケーション属性1のセット F_RSTA1　アプリケーション属性1のリセット F_SETA2　アプリケーション属性2のセット F_RSTA2　アプリケーション属性2のリセット

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	ファイル(lnk)の所有者でない、またはレベル0のユーザでない
E_MACV	アドレス(lnk)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_PAR	パラメータが不正である(attr が不正)
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である。
E_SYSMEM	システムのメモリ領域が不足した

解説

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

指定したファイルのアクセス属性を変更する。

アクセスモードの変更は、ファイルシステムのアクセスレベルが 0 のときは誰でも変更可能であるが、アクセスレベルが 0 でないときはファイルの所有者のプロセスのみ変更可能となる。

すでにオープンされているファイルのアクセスモードを変更した場合、その変更はすでにオープンされているものには影響を与えない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル名変更

tkse_chg_fnm

C言語インタフェース

```
ER ercd = tkse_chg_fnm(LINK *lnk, TC *name);
```

パラメータ

LINK	*lnk	対象ファイル
TC	*name	変更するファイル名(TNULL、または最大ファイル名文字数まで有効)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	ファイル(lnk)の所有者でない、またはレベル0のユーザでない
E_MACV	アドレス(lnk,name)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_FNAME	ファイル名(name)は空または不正である
E_PERM	ファイル(lnk)は削除不可である(削除不可属性がセットされている)
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である。
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したファイルのファイル名を変更する。

ファイル名の変更は、ファイルシステムのアクセスレベルが 0 のときは誰でも変更可能であるが、アクセスレベルが 0 でないときはファイルの所有者のプロセスのみ変更可能となる。

書き込み不可属性、または削除不可属性のセットされているファイルのファイル名の変更はできない。

指定したファイルがリンクファイルのときは、参照先のファイル名およびリンクファイル内に保持されている参照ファイル名の両方が変更される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル日時変更

tkse_chg_ftm

C言語インタフェース

```
ER ercd = tkse_chg_ftm(LINK *lnk, F_TIME *times);
```

パラメータ

LINK	*lnk	対象ファイル
F_TIME	*times	変更する日時
		NULL　現在日時に設定

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	ファイル(lnk)の所有者でない、またはレベル0のユーザでない
E_MACV	アドレス(lnk,times)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である。
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したファイルの保存期限、最新アクセス日時、および最新更新日時を変更する。
F_TIME のそれぞれの値が ≦0 のときはその項目は変更しない。
ファイル日時の変更は、ファイルシステムのアクセスレベルが 0 のときは誰でも変更可能であるが、アクセスレベルが 0 でないときはファイルの所有者のプロセスのみ変更可能となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル情報の取得

tkse_fil_sts

C言語インタフェース

```
ER ercd = tkse_fil_sts(LINK *lnk, TC *name, F_STATE *stat, F_LOCATE *locat);
```

パラメータ

LINK	*lnk	対象ファイル
TC	*name	ファイル名の格納領域(最大ファイル名＋1文字分の領域)
		NULL　格納しない
F_STATE	*stat	ファイル管理情報の格納領域
		NULL　格納しない
F_LOCATE	*locat	ファイル所在情報の格納領域
		NULL　格納しない

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		≧ 0　正常終了(ファイルの参照カウント)

エラーコード

E_MACV	アドレス(lnk,name,stat,locat)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したファイルの情報を取り出す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル情報の取得

tkse_ofl_sts

C言語インタフェース

```
ER ercd = tkse_ofl_sts(W fd, TC *name, F_STATE *stat, F_LOCATE *locat);
```

パラメータ

W	fd	ファイルディスクリプタ
TC	*name	ファイル名の格納領域(最大ファイル名＋1文字分の領域)
		NULL　格納しない
F_STATE	*stat	ファイル管理情報の格納領域
		NULL　格納しない
F_LOCATE		*locat　ファイル所在情報の格納領域
		NULL　格納しない

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		≧ 0　正常終了(ファイルの参照カウント)

エラーコード

E_MACV	アドレス(name,stat,locat)のアクセスは許されていない
E_FD	ファイルディスクリプタは存在していない
E_IO	入出力エラーが発生した
E_SYSMEM	システムのメモリ領域が不足した

解説

オープンしたファイルの情報を取り出す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

リンクファイル情報の取得

tkse_lnk_sts

C言語インタフェース

```
ER ercd = tkse_lnk_sts(LINK *lnk, F_LINK *stat);
```

パラメータ

LINK	*lnk	対象リンクファイル
F_LINK	*stat	リンクファイル情報の格納領域
		NULL　格納しない

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		≧ 0　正常終了(リンクファイルの参照カウント)

エラーコード

E_MACV	アドレス(lnk,stat)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_NOLNK	リンクファイルではない
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したリンクファイルのリンクファイル情報を取り出す。
指定したファイルがリンクファイルでないときはエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

リンクファイルの同期

tkse_syn_lnk

C言語インタフェース

```
ER ercd = tkse_syn_lnk(LINK *lnk, W opt);
```

パラメータ

LINK	*lnk	対象リンクファイル
W	opt	同期属性
		=0　チェックのみ
		≠0　チェックおよび更新

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		≧ 0　正常終了(同期状態)

エラーコード

E_MACV	アドレス(lnk)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOEXS	ファイル(lnk)は存在していない
E_NOFS	ファイル(lnk)またはファイル(lnk)の参照先のファイルの属するファイルシステムは接　続されていない。
E_NOLNK	リンクファイルではない
E_RONLY	ファイル(lnk)は書込不可である、または属するファイルシステムは書込不可である。
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したリンクファイルが保持しているファイル名、生成日時と、参照先のファイルの実際のファイル名、生成日時が一致しているかどうかをチェックする。

opt=0 のときはチェックのみを行い、opt ≠ 0 のときは異なっていた場合は、指定したリンクファイルが保持している情報を参照先のファイルの実際のファイル名、生成日時と一致するように更新する。

リターン値は以下の同期状態となる。

F_SYNC	一致している。
F_DNAME	ファイル名が異なっていた

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

F_DDATE　生成日時が異なっていた
F_DBOTH　ファイル名と生成日時の両方が共に異なっていた

指定したファイルがリンクファイルでないときはエラーとなる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デフォルトアクセスモードの取得

tkse_get_dfm

C言語インタフェース

```
ER ercd = tkse_get_dfm(DA_MODE *mode);
```

パラメータ

DA_MODE	*mode	デフォルトアクセスモードの格納領域

```
typedef struct {
    UH      f_ownacc;       /* 所有者アクセスモード */
    UH      f_grpacc;       /* グループアクセスレベル */
    UH      f_pubacc;       /* 一般アクセスレベル */
    H       f_grpno;        /* グループ番号(0～４) */
    UH      f_gacc[N_GRP];  /* グループアクセスレベル */
} DA_MODE;
```

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(mode)のアクセスは許されていない

解説

デフォルトアクセスモードを取り出す。

デフォルトアクセスモードは、ファイルの生成時にアクセスモードを指定しなかったときに適用されるデフォルトのアクセスモードである。

f_gacc[4] は、ユーザの所属するグループのそれぞれに対して設定されているグループアクセスレベルを示す単なる参照用のデータであり、f_grpacc が実際のグループアクセスレベルとして適用される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デフォルトアクセスモードの設定

tkse_set_dfm

C言語インタフェース

```
ER ercd = tkse_set_dfm(DA_MODE *mode);
```

パラメータ

DA_MODE	*mode	設定するデフォルトアクセスモード

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である(mode の内容が不正)
E_MACV	アドレス(mode)のアクセスは許されていない

解説

デフォルトアクセスモードを設定する。
変更したデフォルトアクセスモードは、全プロセスに対して有効となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステムの接続

tkse_att_fls

C言語インタフェース

```
ER ercd = tkse_att_fls(TC *dev, TC *name, LINK *lnk, UW mode);
```

パラメータ

TC	*dev	デバイス名
TC	*name	接続名(TNULL、または最大接続名文字数まで有効)
LINK	*lnk	接続したファイルシステムのルートファイルのリンクの格納領域 NULL　格納しない
UW	mode	接続モード (FS_SYNC ・ FS_ASYN ・ FS_RONLY) FS_SYNC : 同期書き込み FS_ASYN : 非同期書き込み FS_RONLY : 書き込み禁止

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	論理デバイス(dev)のアクセス権(接続)がない
E_MACV	アドレス(dev,name,lnk)のアクセスは許されていない
E_BUSY	論理デバイス(dev)は既にオープンされている、または接続されている
E_OBJ	接続名(name)は既に存在しているまたは同一のファイルシステム名のファイルシステムが既に接続されている。
E_FNAME	ファイル名(name)は空または不正である
E_IO	入出力エラーが発生した
E_LIMIT	同時接続可能な最大ファイルシステム数を越えた
E_NODEV	デバイス(dev)へのアクセスができない
E_NOEXS	デバイス(dev)は登録されていない、またはブロック型デバイスでない
E_NOMDA	デバイス(dev)のメディアが存在しない
E_ILFMT	標準ファイルシステムの形式ではない
E_SYSMEM	システムのメモリ領域が不足した

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

解説

指定したデバイス上に存在するファイルシステムを指定した接続名でシステムに接続する。

接続名は、接続したファイルシステムのルートファイルを絶対パス名により指定するために使用され、すでに接続済みの接続名と同じであってはいけない。

FS_SYNC（同期書き込み)を指定した場合は、ファイルへの書き込みは書き込みのシステムコールを実行した時点で必ず行われる。

FS_ASYN（非同期書き込み）を指定した場合は、ファイルへの書き込みは書き込みのシステムコールを実行した時点で行われるとは限らない。

FS_RONLY（書き込み禁止）を指定した場合は、ファイルへの書き込みはすべて禁止される。

すでに接続済みのファイルシステムを再度接続しようとしたとき、また、接続しようとしたファイルシステムのファイルシステム名と同一のファイルシステム名を持つファイルシステムがすでに接続されていたときはエラーとなる。

ファイルシステムを接続するには、デバイスに対する接続アクセス権が必要となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステムの切断

tkse_det_fls

C言語インタフェース

```
ER ercd = tkse_det_fls(TC *dev, W eject);
```

パラメータ

TC	*dev	デバイス名
W	eject	イジェクト指定
		＝0　イジェクトしない
		≠0　イジェクトする(イジェクト不可能なデバイスの時は無視)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	論理デバイス(dev)のアクセス権(接続)がない
E_MACV	アドレス(dev)のアクセスは許されていない
E_BUSY	ファイルシステムは使用中である
E_IO	入出力エラーが発生した
E_NOEXS	デバイス(dev)は登録されていない、またはブロック型デバイスでない
E_NOFS	論理デバイス(dev)上の属するファイルシステムは接続されていない
E_NOMDA	デバイスのメディアが存在しない
E_RONLY	ファイルは書込不可である、または属するファイルシステムは書込不可である
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したデバイス上の接続済みのファイルシステムをシステムから切り離す。このとき、メモリ上に一時的に保持されている内容があればファイルシステム上にすべて書き出す。

切断の対象となるファイルシステム内のファイルがオープンされているとき、または作業ファイルとしているプロセスが存在しているときは、切断できない。

ファイルシステムを切断するには、デバイスに対する接続アクセス権が必要となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステムの同期

tkse_syn_fls

C言語インタフェース

```
ER ercd = tkse_syn_fls(void);
```

パラメータ

なし

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_IO	入出力エラーが発生した
E_NOMDA	デバイスのメディアが存在しない
E_RONLY	ファイルは書込不可である、または属するファイルシステムは書込不可である
E_SYSMEM	システムのメモリ領域が不足した

解説

メモリ上に一時的に保持されていた内容をファイルシステム上にすべて書き出し、ファイルシステム全体を矛盾ないように更新する。接続されているすべてのファイルシステムに対して行う。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステム管理情報の取得

tkse_fls_sts

C言語インタフェース

```
ER ercd = tkse_fls_sts(TC *dev, FS_STATE *buff);
```

パラメータ

TC	*dev	デバイス名
FS_STATE	*buff	ファイルシステム管理情報の格納領域

リターンパラメータ

ER	ercd	＜ 0	エラーコード
		＝ 0	正常終了(ファイルシステムは書込可)
		＝ 1	正常終了(ファイルシステムは書込不可)

エラーコード

E_FACV	論理デバイス(dev)のアクセス権(接続)がない
E_MACV	アドレス(dev,buf)のアクセスは許されていない
E_BUSY	論理デバイスは既にオープンされている
E_IO	入出力エラーが発生した
E_NODEV	デバイス(dev)へのアクセスができない
E_NOEXS	デバイス(dev)は登録されていない、またはブロック型デバイスでない
E_NOMDA	デバイス(dev)のメディアが存在しない
E_ILFMT	標準ファイルシステムの形式ではない
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したデバイス上の接続済みのファイルシステムの管理情報を取り出す。
ファイルシステムの管理情報を取り出すためには、デバイスに対する接続アクセス権が必要となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステム情報の変更

tkse_chg_fls

C言語インタフェース

```
ER ercd = tkse_chg_fls(TC *dev, TC *fs_name, TC *fs_locate);
```

パラメータ

TC	*dev	デバイス名 NULL の場合エラーとなる
TC	*fs_name	変更するファイルシステム名 NULL　変更しない
TC	*fs_locate	変更するデバイス所在名 NULL　変更しない

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FACV	論理デバイス(dev)のアクセス権(接続、書込)がない
E_MACV	アドレス(dev,fs_name,fs_locate)のアクセスは許されていない
E_BUSY	論理デバイスは既にオープンされている
E_OBJ	指定したファイルシステム名のファイルシステムは既に存在している(接続されている)
E_FNAME	ファイルシステム名は空、または不正
E_IO	入出力エラーが発生した
E_NODEV	デバイス(dev)へのアクセスができない
E_NOEXS	デバイス(dev)は登録されていない、またはブロック型デバイスでない
E_NOMDA	デバイス(dev)のメディアが存在しない
E_RONLY	ファイルは書込不可である、または属するファイルシステムは書込不可である
E_ILFMT	標準ファイルシステムの形式ではない
E_SYSMEM	システムのメモリ領域が不足した

解説

指定したデバイス上の接続済みのファイルシステムのファイルシステム名およびデバイス所在名を変更する。
ファイルシステム情報を変更するには、デバイスに対する接続アクセス権、および書き込みアクセス権が

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

必要となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

リンクの順次取得

tkse_get_nlk

C言語インタフェース

```
ER ercd = tkse_get_nlk(LINK *lnk);
```

パラメータ

LINK	*lnk	開始ファイルのリンク（入力） 次のファイルのリンクの格納領域（出力）

リターンパラメータ

ER	ercd	＜ 0　エラーコード ≧ 0　正常終了(取り出したファイルの参照カウント)

エラーコード

E_MACV	アドレス(lnk)のアクセスは許されていない
E_IO	入出力エラーが発生した
E_NOFS	ファイル(lnk)の属するファイルシステムは接続されていない
E_NOEXS	ファイル(lnk)より大きなファイル ID を持つファイルは存在していない
E_SYSMEM	システムのメモリ領域が不足した

解説

指定した開始ファイルのファイル ID より大きなファイル ID を持つファイルのうち最小のファイル ID を持つファイルへのリンクを取り出す。

開始ファイルのファイル ID は実際には存在しないファイルでも良い。取り出したファイルのリンクの属性データはすべて 0 となる。

ファイルシステムのルートファイル（ファイル ID ＝ 0）からリンクを順次取得することにより、ファイルシステム上に存在するすべてのファイル（参照カウント ＝ 0 のファイルも含む）へのリンクを取り出すことができる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステムの取得

tkse_lst_fls

C言語インタフェース

```
ER ercd = tkse_lst_fls(F_ATTACH *buff, W cnt);
```

パラメータ

F_ATTACH	*buff		ファイルシステム接続情報の格納領域(配列)
	typedef struct {		
	TC	a_name[L_CONNM];	/* 接続名 */
	TC	dev[L_DEVNM];	/* 論理デバイス名 */
	} F_ATTACH;		
W	cnt	＞ 0	buff の要素数を示す。
		＝ F_GETDEV	デバイス名を取り出す。
		＝ F_GETNAM	接続名を取り出す。

リターンパラメータ

ER	ercd	＜ 0	エラーコード
		≧ 0	正常終了(接続済みのファイルシステム数)
		＝ 1	正常終了(F_GETDEV, F_FETNAM 指定時)

エラーコード

E_MACV	アドレス(buff)のアクセスは許されていない
E_NOFS	ファイルシステムは接続されていない(cnt=－1,－2 の時)
E_PAR	パラメータが不正である(cnt=0,＜－2)

解説

接続済みのファイルシステムの接続名とデバイス名と取り出す。

cnt が ＞ 0 の場合は、接続されているすべてのファイルシステムの接続情報を buff に取り出す。接続されているファイルシステムの数が指定した要素数(cnt)より多い場合は、最初の要素数(cnt)個のみを取り出す。

cntにF_GETDEVが指定された場合は、buff->a_name[] に設定した接続名に対応するデバイス名を buff->dev[] に取り出す。

cnt に F_GETNAM が指定された場合は、buff->dev[] に設定したデバイス名に対応する接続名を

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

buff->a_name[] に取り出す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコードのマップ

tkse_map_rec

C言語インタフェース

```
ER ercd = tkse_map_rec(W fd, W offset, B **addr, W size, W mode);
```

パラメータ

W	fd	ファイルディスクリプタ
W	offset	マップ開始バイトオフセット
B	**addr	マップされたメモリアドレスの格納領域
W	size	マップするバイトサイズ
W	mode	マップモード
		([F_READ] | [F_WRITE] | [F_EXCUTE]) | [F_COMMON ・ F_SYSTEM]
		F_READ　読み込み用マップ
		F_WRITE　書き込み用マップ
		F_EXECUTE　実行用マップ
		F_COMMON　共有メモリ空間にマップ
		F_SYSTEM　システムメモリ空間にマップ

リターンパラメータ

ER	ercd	＜ 0　エラーコード
		＞ 0　正常終了(マップ ID)

エラーコード

E_FD	ファイルディスクリプタは存在していない マップモードとオープンモードが矛盾している
E_REC	現在レコードはリンクレコードである
E_MACV	アドレス(addr)のアクセスは許されていない
E_ENDR	現在レコードは終端レコードである
E_LOCK	現在レコードは他からロックされている
E_IO	入出力エラーが発生した
E_PAR	パラメータが不正である
E_NOSPT	システムの制限によりマップできない

解説

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

オープンしたファイルの現在レコードの offset から size バイトをメモリ空間上にマップする。マップされたレコードの内容はメモリとしてアクセスすることができる。

F_COMMON を指定した場合には、共有メモリ空間上にマップされる。この場合、すべてのプロセスからアクセス可能となる。F_SYSTEM を指定した場合には、システムメモリ空間上にマップされる。この場合、すべてのシステムプロセスからアクセス可能となる。マップしたプロセス自身であっても、一般のユーザプロセスからはアクセスできない。したがって、通常のアプリケーションはこの指定を使用すべきではない。

F_COMMON,F_SYSTEM のいずれも指定しなければ､マップしたプロセスのローカルメモリ空間上にマップされる。この場合、マップしたプロセス以外からアクセスすることはできない。

マップされるアドレスはシステムで決定され、アプリケーション側から指定することはできない。

マップモードの指定はオープンしたモードに矛盾してはいけない。(E_FD)

・F_READ オープンでは、F_WRITE でマップできない。
・F_WRITE オープンでは、F_READ および F_EXCUTE でマップできない。

リンクレコードはマップできない。(E_REC)

マップ中は、次の操作は禁止され E_BUSY となる。

・del_rec マップしているレコードを対象とする場合。
・wri_rec マップしているレコードを対象とする場合。
・trc_rec マップしているレコードを対象とする場合。
・xch_fil マップしているレコードを含むファイルを対象とする場合。

次の場合には、マップできないという制限がある。

・リムーバブルメディア上のファイルシステム
・論理ブロックサイズが 4KB 未満のファイルシステム

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

レコードのアンマップ

tkse_ump_rec

C言語インタフェース

```
ER ercd = tkse_ump_rec(W fd, W mapid);
```

パラメータ

W	fd	ファイルディスクリプタ
W	mapid	マップID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FD	ファイルディスクリプタは存在していない
E_NOEXS	マップIDは存在していない
E_IO	入出力エラーが発生した

解説

オープンしているファイルのマップIDで指定したマップを解除する。ファイルのクローズ時にはオープンしたプロセスで設定したマップは解除される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイルシステム接続モードの変更

tkse_chg_fsm

C言語インタフェース

```
ER ercd = tkse_chg_fsm(TC *dev, UW mode);
```

パラメータ

TC	*dev	論理デバイス名
UW	mode	接続モード (att_fls の mode と同じ)
		(FS_SYNC ・ FS_ASYN ・ FS_RONLY)
		FS_SYNC 同期
		FS_ASYN 非同期
		FS_RONLY 書き込み禁止

リターンパラメータ

ER	ercd	＜ 0 エラーコード
		≧ 0 正常終了(変更前の接続モード)

エラーコード

E_FACV	論理デバイス(dev)に接続アクセス権がない
E_MACV	アドレス(dev)のアクセスが許されていない
E_IO	入出力エラーが発生した
E_PAR	パラメータが不正である
E_NOFS	論理デバイス(dev)は、ファイルシステムとして接続されていない

解説

dev のデバイスの接続状態を mode で指定した接続モードに変更する。dev はすでに接続されているデバイスでなければならない。リターン値に変更前の接続モードを返す。

接続モードを変更するためには、dev に対する接続アクセス権が必要である。

接続モード変更前が書き込み可であった場合、書き込み禁止に変更しても変更前に行われた tkse_map_rec() による書き込みモードのレコードマップはそのまま有効となり、書き込みも行われる。F_WRITE または F_UPDATE でオープンしているファイルがあるとき、接続モードを書き込み禁止にすると、そのファイルに対する tkse_wri_rec() 等の書き込みが E_RONLY となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ファイル単位の同期

tkse_syn_fil

C言語インタフェース

```
ER ercd = tkse_syn_fil(W fd);
```

パラメータ

W	fd	ファイルディスクリプタ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_FD	ファイルディスクリプタは存在していない

解説

指定されたオープン中のファイルに関して、一時的にメモリに保持されている内容をディスクに書き出す。レコードマップ(tkse_rec_map)上による書き込みも同様にディスクに書き出される。

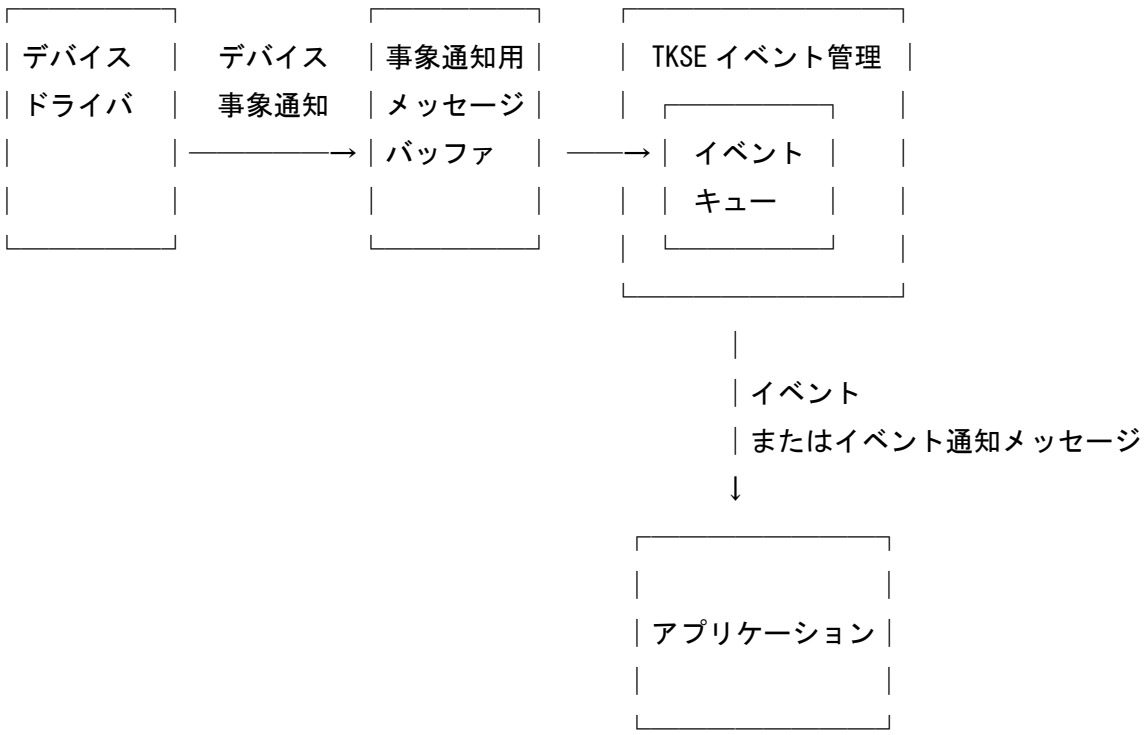
Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.8 イベント管理

4.8.1 概要

Standard Extension のイベント管理機能は、各種デバイスからの事象の通知を「イベント」という形で統一的に取り扱う。イベント管理機能の主な目的は、インタラクティブなヒューマンインタフェースを実現することである。よって、イベント管理の対象となるデバイスは、キーボードやポインティングデバイスを主として想定しているが、その他のデバイスも対応可能である。

イベント管理の仕組みを以下に示す。



[図 11] イベント管理の位置付け

各種デバイスからの事象は、T-Kernel のデバイス事象通知の機能を用いてイベント管理に伝えられる。イベント管理は、デバイスからの事象通知を基にイベントを生成し、イベントキューに順次格納する。

イベント管理とデバイスとのインタフェースは、原則としてデバイス事象通知のみとする。イベント管理から直接デバイスを制御する機能は提供しない。また、イベント管理が特定のデバイスに依存することはない。

アプリケーションはイベントキューからイベントを順次取り出し、それに対応するアクションを実行する。またアプリケーションは、イベント発生の通知をプロセス間メッセージ機能を用いて取得することも可能である。その際、イベント発生時に、プロセス間メッセージ機能のイベント発生の通知（tkse_brk_msg）が呼び出される。

ある時点では特定の１つのプロセスのみがイベント管理機能を使用してイベントを取り出す、というルールを

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

前提とする。アプリケーションはこのルールに従った動作をしなくてはいけない。イベント管理機能自体では、このルールを保証する機構は提供していない。

4.8.2 イベントの種類

イベントは、0 ～ 15 のタイプ番号で区別される。イベントタイプとして以下のものを定義する。

・タイプ番号 0　　ヌルイベント（ EV_NULL ）
対象とするイベントが発生していないことを示す擬似的なイベント。

・タイプ番号 1　　ボタンダウンイベント（ EV_BUTDWN ）
デバイスのボタンが押された時に発生する。

・タイプ番号 2　　ボタンアップイベント（ EV_BUTUP ）
デバイスのボタンが離された時に発生する。

・タイプ番号 3　　キーダウンイベント（ EV_KEYDWN ）
キーボードのキーが押された時に発生する。

・タイプ番号 4　　キーアップイベント（ EV_KEYUP ）
キーボードのキーが離された時に発生する。

・タイプ番号 5　　自動リピートキーイベント（ EV_AUTKEY ）
自動リピートの対象となるキーが押され続けていた時に周期的に発生する。自動リピートの対象となるキーを押してから、最初に自動リピートキーイベントが発生するまでの時間(オフセット)、およびその後の発生間隔(インターバル)は任意に設定可能である。

・タイプ番号 6　　デバイスイベント（ EV_DEVICE ）
デバイスのある種の操作に伴って発生する汎用的なイベントであり、その内容はデバイスに依存する。ディスク等に取り外し可能メディアを装着した場合などで、このイベントが発生する。

・タイプ番号 7　　拡張デバイスイベント（ EV_EXDEV ）
デバイスイベント（ EV_DEVICE)に、拡張情報を付加したイベント。
ucode（ユビキタス ID）などの拡張情報を伴うデバイスイベントに対し使用する。

・タイプ番号 8～15　アプリケーションイベント（ EV_APPL1 ～ EV_APPL8 ）
アプリケーションによって自由に使用可能なイベントである。アプリケーション間の通信機能として使用することができる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

また、各イベントタイプに対応したタイプマスクが定義されている。このマスクにより、対象とするイベントのタイプを指定することができる。タイプマスクはビット対応となっており、"1" のビットに対応するタイプのイベントを対象とする。ただし、ヌルイベントに対しては、その性質上マスクは定義されない。

[表 2] イベントタイプ番号とイベントタイプマスク

イベント	タイプ番号	タイプマスク
EV_NULL	0	------
EV_BUTDWN	1	EM_BUTDWN (0x0001)
EV_BUTUP	2	EM_BUTUP (0x0002)
EV_KEYDWN	3	EM_KEYDWN (0x0004)
EV_KEYUP	4	EM_KEYUP (0x0008)
EV_AUTKEY	5	EM_AUTKEY (0x0010)
EV_DEVICE	6	EM_DEVICE (0x0020)
EV_EXDEV	7	EM_EXDEV (0x0040)
EV_APPL1	8	EM_APPL1 (0x0080)
:	:	: :
EV_APPL8	15	EM_APPL8 (0x4000)

なお、タイプマスクとして以下の特殊なマスクを定義する。

```
EM_NULL 0x0000
EM_ALL  0x7fff
```

4.8.3 デバイス事象通知からのイベント生成

イベント管理は、デバイス事象通知の事象タイプ（TDEvtTyp）からイベント（9.5 章の EVENT 構造体）を生成し、これをイベントキューに格納する。事象タイプには 4 タイプあり、それぞれ以下のようにイベントタイプを決定する。

(1) 基本事象（TDEvtTyp ＝ 0x0001 ～ 0x002F）

(A) キーボード・ポインティングデバイス以外の基本事象：

規定済みの事象：

TDE_unknown	0x00	未定義
TDE_MOUNT	0x01	メディア挿入

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

TDE_EJECT	0x02	メディア排出
TDE_ILLMOUNT	0x03	メディア不正挿入
TDE_ILLEJECT	0x04	メディア不正排出
TDE_REMOUNT	0x05	メディア再挿入
TDE_CARDBATLOW	0x06	カードバッテリ残存警告
TDE_CARDBATFAIL	0x07	カードバッテリ異常
TDE_REQEJECT	0x08	メディア排出要求

→ これらの事象はデバイスイベント（EV_DEVICE）としてイベントキューに格納される。
また、下記 キーボード・ポインティングデバイス以外の基本事象もデバイスイベントとなる。

(B) キーボード・ポインティングデバイスなどの基本事象：

規定済みの事象：

TDE_PDBUT	0x11	PD ボタン状態の変化
TDE_PDMOVE	0x12	PD 位置移動
TDE_PDSTATE	0x13	PD の状態変化
TDE_PDEXT	0x14	PD 拡張事象
TDE_KEYDOWN	0x21	キーダウン
TDE_KEYUP	0x22	キーアップ
TDE_KEYMETA	0x23	メタキー状態の変化

→ これら事象は、ボタンダウン（EV_BUTDWN)、ボタンアップ（EV_BUTUP)、キーダウン（EV_KEYDWN)、
キーアップ（EV_KEYUP）としてイベントキューに格納される。

(2) システム事象（TDEvtTyp ＝ 0x0030 ～ 0x007F）

規定済みの事象

TDE_POWEROFF	0x31	電源スイッチオフ
TDE_POWERLOW	0x32	電源残量警告
TDE_POWERFAIL	0x33	電源異常
TDE_POWERSUS	0x34	自動サスペンド
TDE_POWERUPTM	0x35	時刻更新
TDE_CKPWON	0x41	自動電源オン通知

・システム事象は、イベントキューに格納されない。

(3) 拡張情報付き事象（TDEvtTyp ＝ 0x0080 ～ 0x00FF）

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

規定済みの事象
→ 特に無し。

・拡張情報付き事象は、拡張デバイスイベント（EV_EXDEV）としてイベントキューに格納される。

(4) ユーザ定義事象（TDEvtTyp ＝ 0x0100 ～ 0xFFFF）

規定済みの事象
→ 特に無し。

・ユーザ定義事象は、イベントキューに格納されない。また、原則としてイベント管理では使用しない。

4.8.4 イベントキューとイベントの優先度

イベントキューは、イベント管理に 1 つだけ用意されているイベント格納用のキューであり、イベントが発生順に格納される。キューに空きが無い場合は、新規に発生したイベント、即ち、一番新しいイベントはキューに入れられず、捨てられることになる。

イベントキューに格納されるイベントはシステムイベントマスクにより制限される。システムイベントマスクの "1" のビットに対応するタイプのイベントのみがイベントキューに入り、"0" のビットに対応するタイプのイベントは無視され、キューには格納されない。

システムのスタートアップ時点では、イベントキューは空、システムイベントマスクは 0 となっており、イベント管理機能は事実上動作していない状態であるため、イベントを受け取るためには必ずシステムイベントマスクを適当な値に設定する必要がある。

イベントには、そのタイプに応じた以下に示す優先度が付けられており、高い優先度のイベントよりイベントキューから取り出される。同一優先度の場合は、発生順に取り出される。

（ 1: 最高優先度 ～ 6:最低優先度 ）

1. EV_APPL1～4
2. EV_BUTDWN, EV_BUTUP, EV_KEYDWN, EV_KEYUP
3. EV_AUTKEY
4. EV_DEVICE, EV_EXDEV
5. EV_APPL5～8
6. EV_NULL

ヌルイベント(EV_NULL)、および自動リピートイベント(EV_AUTKEY) は、実際にはイベントキューには入れられず、イベントの取り出し要求時に条件が成立したときに自動的に生成される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.8.5 キーボードからのイベント

(1) 通常キーとメタキー

キーは、押したり離したりした時にイベントを発生する通常キーと、イベントを発生しないメタキーに分けられる。メタキーは、シフトキーなどの他の通常キーと組み合わせて使用されるキーである。
通常キーはその押し離しにより以下のイベントを発生する。

・キーダウンイベント（ EV_KEYDWN ）: キーボードのキーが押された時に発生する。
・キーアップイベント（ EV_KEYUP ）: キーボードのキーが離された時に発生する。
・自動リピートイベント（ EV_AUTKEY ）: 対象となるキーが押され続けていた時に周期的に発生する。

(2) 自動リピートキー

自動リピートキーイベント(EV_AUTKEY) は、自動リピートの対象キーが押し続けられた場合に発生する。
1 つのキーを押し続けている状態で、さらに別のキーを押し続けた場合は、最後に押したキーの自動リピートキーイベントのみが発生する。
自動リピートの対象キーはイベントを発生しないメタキーを除いて任意に設定可能であり、システム立ち上げ時には、メタキーを除いたすべてのキーを自動リピートの対象とする。
押してから最初に発生するまでの時間(オフセット時間)と、その後の発生間隔(インターバル時間)の設定／取り出し用のシステムコールが提供されている。 このとき間はミリ秒単位である。但し、自動リピートキーイベントの発生間隔は実装依存であり､オフセット時間とインターバル時間は､イベント発生時刻の単位へ丸められる。

4.8.6 キーイベントの文字コード

キーイベントでは、キーの物理的位置を示すキートップコードと、エンコードされた文字コードを戻す。
キートップコードは、キーの物理的位置に応じた固定的な 8 ビットのコード（ 0 ～255 ）である。
文字コードは、キートップコードとメタキーの状態によりエンコードされたコードである。エンコードは、イベント管理より下位のドライバまたはハードウェア等により行われ、イベント管理では感知しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.8.7 ポインティングデバイスからのイベント

ポインティングデバイスは、スクリーン上に表示されているオブジェクトを選択するために使用され、その現在位置として、スクリーンの解像度に対応したレンジの絶対座標値を持つ。 絶対座標値とは、スクリーンの左上の点を（ 0, 0)とし、スクリーン上の 1 ピクセルを単位とした座標値である。

ポイントデバイスは、そのデバイスのボタンの押し離しににより以下のイベントを発生する。

・ボタンダウンイベント（ EV_BUTDWN ）: デバイスのボタンが押された時に発生する。
・ボタンアップイベント（ EV_BUTUP ）: デバイスのボタンが離された時に発生する。

4.8.8 ポインティングデバイスの動作タイプ

ポインティングデバイスは、その動作方式から以下の 2 種類に大きく分類される。

(1) 絶対動作タイプ
電子ペン等のタブレットタイプのもので、ポインティングデバイスの物理的な位置により、絶対的に座標値が決定されるもの。

(2) 相対動作タイプ(差動タイプ):
マウス等で、ポインティングデバイスの物理的な位置の移動により、相対的に座標値が決定されるもの。

相対動作タイプにおける絶対座標値の計算は、イベント管理より下位のドライバまたはハードウェア等により行われ、イベント管理では感知しない。

4.8.9 ホイール対応

ホイールマウスのホイールの回転は、 自動リピートキー（EV_AUTKEY）として扱われる。ただし、イベントの優先度は本来のキーリピートによる EV_AUTKEY とは異なり、ホイールによる EV_AUTKEY は最低優先度(EV_NULL よりは上)とする。

ホイールの回転による EV_KEYDWN, EV_KEYUP イベントは発生しない。

ホイールのイベントは、下記の条件のいずれかが成立したとき、イベントキューから自動的に消滅する。

(1) ホイールを回転させてから 300ms 以内にアプリケーションがホイールイベントを取り出さなかったとき。
(2) ホイール以外のイベント(EV_NULL を除く）をアプリケーションが取り出したとき。

なお、アプリケーションが取り出すまでの間に、ホイールが複数回回転させられたときは、その合計量が 1 回のイベントとして扱われる。ただし、回転方向が反転した場合は、それ以前の回転量は捨てられる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.8.10 イベントの構造

イベントは以下に示す構造体で定義される。

```
typedef struct {
        W       type;       /* イベントタイプ */
        UW      time;       /* イベント発生時刻 */
        PNT     pos;        /* イベント発生時のポインティングデバイス位置 */
        EVDATA  data;       /* イベントの固有データ */
        UW      stat;       /* メタキー、ボタン状態 */
        UB      exdat[16]; /* 拡張情報 */
} EVENT;
```

(1) type

イベントのタイプを示す 0 ～ 15 の値である。

(2) time

イベント発生時刻を示すミリ秒単位の相対的な時刻であり、この値はイベントの発生順序、発生間隔を示しているもので絶対時刻としての意味は持たない。

イベント発生時刻は、イベントタイマにより計測される。イベントタイマは T-Kernel のシステム時刻管理機能を用いて実装される。イベントタイマの分解能はその実装に依存する。

(3) pos

イベント発生時点のポインティングデバイスの座標位置を、スクリーンの左上を (0, 0) とした絶対座標値で示すもので、以下の PNT 型の値である。

```
typedef struct point {
        H       x;      /* 水平座標値 */
        H       y;      /* 垂直座標値 */
} PNT;
```

なお、アプリーションイベントでの pos の意味はイベントの定義に依存する。

(4) data

イベントの固有データであり、イベントのタイプに依存した内容である。

```
typedef union {
        struct {        /* EV_KEYUP,EV_KEYDWN,EV_AUTKEY */
                UH      keytop; /* キートップコード */
                TC      code;   /* 文字コード */
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
        } key;
        struct {        /* EV_DEVICE,EV_EXDEV */
                H       kind;   /* デバイスイベント種別 */
                H       devno;  /* デバイス番号 */
        } dev;
        W       info;   /* その他のイベント用データ */
} EVDATA;
```

EV_KEYDWN, EV_KEYUP, EV_AUTKEY の場合は、key が適用され、キーの物理的な位置を示すキートップコードと、エンコードされた文字コードからなる。

ホイールの回転により発生した EV_AUTKEY イベントの内容は以下のようになる。

data.key.keytop ：	0x8000 + 回転量	
	回転量 ＞ 0	ホイールを手前に回転
	回転量 ＜ 0	ホイールを奥に回転
data.key.code ：	KC_SS_D	ホイールを手前に回転
	KC_SS_U	ホイールを奥に回転

EV_DEVICE の場合は、dev が適用され、デバイスイベントの種別(kind)とイベントが発生したデバイスを示すデバイス番号(devno)ｇが設定される。イベントの種別は以下が定義済みである。

kind ＝	DE_unknown	0	-- 未定義
	DE_MOUNT	0x01	-- メディア挿入
	DE_EJECT	0x02	-- メディア排出
	DE_ILLMOUNT	0x03	-- メディア不正挿入
	DE_ILLEJECT	0x04	-- メディア不正排出
	DE_REMOUNT	0x05	-- メディア再挿入
	DE_BATLOW	0x06	-- バッテリ残量警告
	DE_BATFAIL	0x07	-- バッテリ異常
	DE_REQEJECT	0x08	-- メディア排出要求
		0x09～	-- 予約
		0x7F	

また EV_EXDEV の場合も、dev が適用され、デバイスイベントの種別(kind)とイベントが発生したデバイスを示すデバイス番号(devno)が設定される。通常、デバイスイベントの種別(kind)は、デバイス事象通知の事象タイプがそのまま設定される。

kind ＝	XXXXXXXXXX	0x80～	-- 拡張デバイスイベント種別(＝ デバイス事象タイプ)
		0xFF	

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

EV_NULL, EV_BUTDWN, EV_BUTUP の場合は、この data は使用されず、info が常に 0 となる。
アプリーションイベント(EV_APPL1～8) の場合は、イベントの定義に依存した内容となる。

(5) stat

イベント発生時点のメタキーなど、単独ではイベントを発生しないキーやボタンの状態をビット対応で示すものである。各ビットは、"0" が離されている(OFF)状態、"1" が押されている(ON)状態を意味する。各ビットの意味は、以下のように定義される。

・ bit 0～ 1 (2bit) PD 基本ボタン :
　ポインティングデバイスのメインボタンとサブボタンの状態を表す。

・ bit 2～20 (19bit) メタキー　　:
　キーボードのメタキーの状態を表す。
　各メタキーとのビットの対応はイベント管理では定義しない。

・ bit 21　 (1bit) PD 種別　　:
　ポインティングデバイスの種別を表す。

・ bit 22～23(2bit) PD 状態　　:
　ポインティングデバイスの状態を表す。

・ bit 24～31(8bit) PD 拡張ボタン :
　ポインティングデバイスの拡張ボタンの状態を表す。
　各ボタンとビットの対応はイベント管理では定義しない。

(6) exdat

イベントタイプ (type) が拡張デバイスイベント (EV_EXDEV) である場合に有効である。
主としてデバイスイベント通知から渡される､ucode などの拡張情報(16 バイト)を格納するために使用する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.8.11 システムコール

イベントの取得

tkse_get_evt

C言語インタフェース

```
ER ercd = tkse_get_evt(W t_mask, EVENT* evt, W opt);
```

パラメータ

W	t_mask	対象イベントタイプのマスク
EVENT*	evt	取得したイベントの格納領域
W	opt	取得属性 (CLR ・ NOCLR) CLR　イベントキューから取り除く。 NOCLR　イベントキューから取り除かない。

リターンパラメータ

ER	ercd	≧ 0	正常終了(得られたイベントのタイプ)
		＜ 0	エラーコード

エラーコード

E_MACV	アドレス(evt)のアクセスは許されていない
E_IO	入出力エラーが発生した(何らかのデバイスエラーが発生した)
E_PAR	パラメータが不正である(t_mask≦0、opt が不正)
E_SYSMEM	システムのメモリ領域が不足した

解説

t_mask で指定したタイプのイベントをイベントキューから取り出し、evt で指定した領域に格納する。指定したタイプのイベントが発生していないときは、EV_NULL が取り出される。

イベントをイベントキューから取り出した際の処理は、opt で指定される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントの発生

tkse_put_evt

C言語インタフェース

```
ER ercd = tkse_put_evt(EVENT* evt, W opt);
```

パラメータ

EVENT*	evt	発生するイベント
W	opt	発生属性 (EP_NONE ・ EP_ALL ・ ([EP_POS] | [EP_STAT] | [EP_TIME]))

EP_NONE　time, pos, stat は evt の内容のままとする。
EP_ALL　time, pos, stat のすべてを設定する。
EP_POS　pos に現在のポインティングデバイスの位置を設定する。
EP_STAT　stat に現在のメタキー/ PD ボタン状態を設定する。
EP_TIME　time に現在のイベントタイマの値を設定する。

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(evt)のアクセスは許されていない
E_IO	入出力エラーが発生した(何らかのデバイスエラーが発生した)
E_PAR	パラメータが不正である(イベントのタイプが不正、opt が不正)
E_SYSMEM	システムのメモリ領域が不足した

解説

evt で指定したイベントを発生させてイベントキューに入れる。イベントキューに空きがない場合、または、EV_NULL および EV_AUTKEY が指定されたときはエラーとなる。また、システムイベントマスクで対象外のイベントは実際には発生せず無視される。

発生したイベントは time の値にかかわらず、tkse_put_evt()を実行したときに発生したものとみなされ、常にイベントキューの最後に入れられる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントのクリア

tkse_clr_evt

C言語インタフェース

```
ER ercd = tkse_clr_evt(W t_mask, W last_mask);
```

パラメータ

W	t_mask	クリア対象イベントタイプマスク	
		EM_ALL	全イベントタイプ
W	last_mask	クリア終了イベントタイプマスク	
		EM_ALL	イベントを１つだけクリア
		EM_NULL	対象はイベントキューの最後まで

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である(t_mask≦0、last_mask<0)

解説

発生済みのイベントをクリアする。

イベントキューに入っているイベントのうち、t_mask で指定したタイプのイベントを last_mask で指定したタイプのイベントの直前までクリアする。last_mask で指定したタイプのイベントはクリアされないが、last_mask ＝ EM_ALL のときは、特別にイベントを１つだけクリアすることを意味する。また、last_mask ＝ EM_NULL のときはイベントキューの最後までが対象となる。

t_mask と last_mask の組み合わせの例を以下に示す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

[表 3] t_mask と last_mask の組み合わせ

t_mask	last_mask	動作
EM_ALL	EM_NULL	すべてのイベントをクリア
--	EM_ALL	t_mask で指定したイベントを 1 つだけクリア
EM_ALL	EM_ALL	先頭のイベントを 1 つだけクリア

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントタイマ値の取得

tkse_get_etm

C言語インタフェース

```
RR ercd = tkse_get_etm(UW* time);
```

パラメータ

UW	*time	イベントタイマ値の格納領域

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(time)のアクセスは許されていない

解説

イベントタイマの現在の値を取り出す。
イベントタイマはミリ秒単位の相対的な時間であるが、実際の分解能は実装に依存する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

システムイベントマスクの変更

tkse_chg_emk

C言語インタフェース

```
ER o_mask = tkse_chg_emk(W mask);
```

パラメータ

W	mask	設定するシステムイベントマスク < 0　変更しない(現在のシステムイベントマスクの取得)

リターンパラメータ

ER	o_mask	変更前のシステムイベントマスク

解説

システムイベントマスクを mask で指定した値に変更し、変更前のシステムイベントマスクの値を戻値として返す。

mask < 0 のときはシステムイベントマスクを変更せず、現在のシステムイベントマスクの値を戻値として返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

イベントメッセージ要求

tkse_req_evt

C言語インタフェース

```
ER ercd = tkse_req_evt(W t_mask);
```

パラメータ

W	t_mask	対象イベントタイプマスク

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である
E_LIMIT	イベントメッセージ要求の登録数がシステムの制限を超えた

解説

t_mask で指定したタイプのイベントが発生した場合に、そのイベントをメッセージとして自プロセスに送信することを要求する。ただし、EV_NULL、EV_AUTKEY はメッセージとして送信することはできない。
また、メッセージとして送信されても、そのイベントはイベントキューから取り除かれない。

t_mask に EM_NULL を指定することにより、イベントメッセージの要求が解除される。また、プロセスが終了したときにも自動的に解除される。

```
struct {
        W       msg_type;       /* メッセージタイプ = MS_SYS5 */
        W       msg_size;       /* メッセージサイズ */
        EVENT   evt;            /* イベント */
        VB      info[];         /* 追加情報 */
}
```

msg_type は MS_SYS5 固定となる。
発生したイベントは evt に格納される。また、イベントの種類によって追加情報が info に格納される場合がある。したがって msg_size は最低で sizeof(EVENT)となり、追加情報があればそのサイズ分大きくなる。追加

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

情報はその内容によりサイズが異なる。

デバイスイベント(EV_DEVICE)の場合は常に追加情報が付加される。追加情報は、デバイスドライバからの事象通知の内容となる。

tkse_rcv_msg()によりメッセージを受信したときの送信プロセス ID は、tkse_put_evt()により発生したイベントの場合はtkse_put_evt()を呼び出したプロセスの ID となる。その他の場合は ID ＝ 1 (初期プロセス)となる。

プロセスのメッセージキューが一杯で送信できなかった場合には、そのイベントメッセージは単に捨てられる。
イベントメッセージを同時に要求できるプロセスの最大数は、システムで制限される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

最終イベント発生からの経過時間の取得

tkse_las_evt

C言語インタフェース

```
ER ercd = tkse_las_evt(W t_mask);
```

パラメータ

W	t_mask		対象イベントタイプマスク

リターンパラメータ

ER	ercd	≧ 0	正常終了(最終イベント発生からの経過時間)
		< 0	エラーコード

エラーコード

E_PAR	パラメータが不正である

解説

t_mask で指定したタイプのイベントの内、最後に発生したイベントから現在までの経過時間をミリ秒単位で返す。

t_mask に EM_BUTDWN または EM_BUTUP が指定された場合、ポインタの移動およびメニューボタンの操作もイベント発生と扱う。またEM_KEYDWNまたはEM_KEYUPが指定された場合、メタキーの状態変化もイベント発生と扱う。

t_mask に EM_NULL を指定することで、すべてのタイプのイベントの最終発生時刻に現在時刻がセットされる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

自動リピート対象キーの設定

tkse_set_krm

C言語インタフェース

```
ER ercd = tkse_set_krm(KeyMap keymap);
```

パラメータ

KeyMap	keymap	自動リピート対象キーマップ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(keymap)のアクセスは許されていない

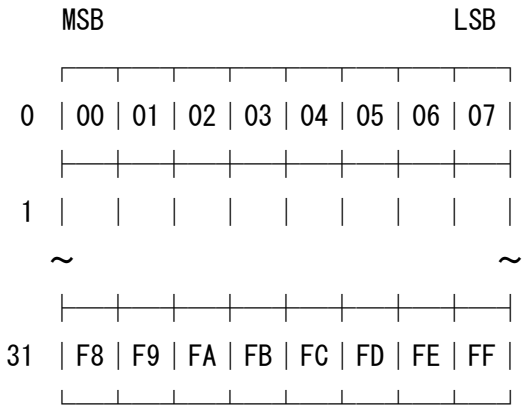
解説

自動リピートの対象となるキーを keymap で指定した値に設定する。
keymap は一つのキー（キートップコード）に 1 ビットを対応させた配列であり、以下のように定義される。

```
typedef UB      KeyMap[KEYMAX/8];
```

KEYMAX はキートップコードの最大値である。keymap の構造は以下のようになる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.



[図 12] keymap 構造

指定したキーマップの "1" のビットに対応するキーを自動リピートの対象とし、"0" のビットに対応するキーを自動リピートの対象としない。

イベントを発生しないメタキーは自動リピートの対象とならず、自動リピート対象キーの設定は無視される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

自動リピート対象キーの取得

tkse_get_krm

C言語インタフェース

```
ER ercd = tkse_get_krm(KeyMap keymap);
```

パラメータ

KeyMap	keymap	自動リピート対象キーマップの格納領域

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(keymap)のアクセスは許されていない

解説

自動リピートの対象となるキーを取り出し、keymap で指定した領域に格納する。

取り出したキーマップの "1" のビットに対応するキーが自動リピートの対象となる。"0" のビットに対応するキーは自動リピートの対象ではない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

自動リピート間隔の設定

tkse_set_krp

C言語インタフェース

```
ER ercd = tkse_set_krp(W offset, W interval);
```

パラメータ

W	offset	自動リピート最初の発生までの時間(ミリ秒)
W	interval	自動リピート発生間隔(ミリ秒)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である(offset≦0,interval≦0)

解説

自動リピートキーイベント(EV_AUTKEY)の発生までの時間、および間隔を offset および interval で指定した値に設定する。

設定する時間はミリ秒単位であるが、実際の分解能は実装に依存する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

自動リピート間隔の取得

tkse_get_krp

C言語インタフェース

```
ER ercd = tkse_get_krp(W* offset, W* interval);
```

パラメータ

W	*offset	自動リピート最初の発生までの時間(ミリ秒)の格納領域
W	*interval	自動リピート発生間隔(ミリ秒)の格納領域

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(offset,interval)のアクセスは許されていない

解説

自動リピートキーイベント(EV_AUTKEY)の発生までの時間、および間隔を取り出し、offset および interval で指定した領域に格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

PD 位置の取得

tkse_get_pdp

C 言語インタフェース

```
ER ercd = tkse_get_pdp(PNT* pos);
```

パラメータ

PNT*	pos	PD 位置の格納領域

```
typedef struct point {
        H    x; /* 水平座標値 */
        H    y; /* 垂直座標値 */
} PNT;
```

リターンパラメータ

ER	ercd	≧ 0	発生しているイベントのタイプ(正常終了)
		＜ 0	エラーコード

エラーコード

E_MACV	アドレス(pos)のアクセスは許されていない

解説

現在のポインティングデバイスの位置を絶対座標値で取り出す。イベントキューの内容は変化しない。

戻値として同時に発生しているイベントのタイプを返す。何のイベントも発生していないときは EV_NULL を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.9 デバイス管理

4.9.1 概要

Standard Extension のデバイス管理機能は、T-Kernel のデバイス管理機能によりシステムに登録されたデバイスを操作するための機能を提供する。

Standard Extension のデバイス管理機能は、デバイスの登録機能は持たない。

4.9.2 デバイスの基本概念

(1) デバイス名 (UB*型)

デバイス名は最大 8 文字の文字列で、次の要素により構成される。

```
#define L_DEVNM 8 /* デバイス名の長さ */
```

・種別

デバイスの種別を示す名前
使用可能な文字は a～ z, A～ Z

・ユニット

物理的なデバイスを示す英字 1 文字
a～ z でユニットごとに a から順に割り当てる

・サブユニット

論理的なデバイスを示す数字最大 3 文字
0～ 254 でサブユニットごとに 0 から順に割り当てる

デバイス名は、種別＋ユニット＋サブユニットの形式で表すが、ユニット、サブユニットはデバイスによっては存在しない場合もあり、その場合はそれぞれのフィールドは存在しない。

特に、種別＋ユニットの形式を物理デバイス名と呼ぶ。また、種別＋ユニット＋サブユニットを論理デバイス名と呼び物理デバイス名と区別することがある。サブユニットがない場合は、物理デバイス名と論理デバイス名は同じになる。単にデバイス名と言ったときは、論理デバイス名を指す。

サブユニットは、一般的にはハードディスクの区画(パーティション)を表すが、それ以外でも論理的なデバイスを表すために使用できる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

（例）　hda　　ハードディスク(ディスク全体)
　　hda0　　ハードディスク(先頭の区画)
　　fda　　フロッピーディスク
　　rsa　　シリアルポート
　　kbpd　　キーボード／ポインティングデバイス

(2) デバイス ID(ID 型)

デバイス ID はデバイスを登録する際に、それぞれのデバイスに一意に割り当てられる ID である。デバイス ID は 0 より大きい整数値である。デバイス ID は物理デバイスごとに割り当てられ、論理デバイスのデバイス ID は物理デバイスに割り当てられたデバイス ID にサブユニット番号+1(1～ 255)を加えたものとなる。

例として、登録時に割り当てられたデバイス ID を devid としたとき

・devid　　物理デバイス
・devid + n+1　　n 番目のサブユニット(論理デバイス)

となる。

(例)　hda　devid　　ハードディスク全体
　　hda0　devid + 1　ハードディスクの先頭の区画
　　hda1　devid + 2　ハードディスクの２番目の区画

(3) デバイス属性 (ATR 型)

各デバイスの特徴を表しデバイスの種類分けを行うため、デバイス属性を次のように定義する。

```
IIII IIII IIII IIII PRxx xxxx KKKK KKKK
```

上位 16 ビットは、デバイス依存属性で、デバイスごとに定義される。下位 16 ビットは、標準属性で、下記のように定義される。

```
#define TD_PROTECT        0x8000    /* P: 書込み禁止 */
#define TD_REMOVABLE      0x4000    /* R: メディアの取り外し可能 */
#define TD_DEVKIND        0x00ff    /* K: デバイス/メディア種別 */
#define TD_DEVTYPE        0x00f0    /* デバイスタイプ */

/* デバイスタイプ */
#define TDK_UNDEF         0x0000    /* 未定義/不明 */
#define TDK_DISK          0x0010    /* ディスクデバイス */

/* ディスク種別 */
#define TDK_DISK_UNDEF    0x0010    /* その他のディスク */
#define TDK_DISK_RAM      0x0011    /* RAM ディスク */
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

```
#define TDK_DISK_ROM      0x0012    /* ROM ディスク */
#define TDK_DISK_FLA      0x0013    /* Flash ROM、その他のシリコンディスク */
#define TDK_DISK_FD       0x0014    /* フロッピーディスク */
#define TDK_DISK_HD       0x0015    /* ハードディスク */
#define TDK_DISK_CDROM    0x0016    /* CD-ROM */
```

上記以外のデバイスタイプは、デバイスドライバ仕様書で別途定義される。未定義のデバイスは、未定義 TDK_UNDEF とする。なお、デバイスタイプは、アプリケーションがデバイス/メディアの種類によって処理内容を変える必要がある場合など、デバイスを利用する側で区別が必要となる場合に定義する。特に明確な区別をする必要がないデバイスに関しては、デバイスタイプを割り当てる必要はない。

デバイス依存属性については、各デバイスドライバの仕様書を参照のこと。

(4) デバイスディスクリプタ (ID 型)

デバイスディスクリプタはデバイスをアクセスするための識別子である。デバイスをオープンした際に、デバイスディスクリプタが新規に割り当てられる。

デバイスディスクリプタは各プロセスに所属する。デバイスディスクリプタを使用した操作は、そのデバイスディスクリプタを割り当てられたプロセスからのみ行うことができる。

デバイスディスクリプタは 0 より大きい整数値である。

(5) リクエスト ID (ID 型)

デバイスに対して非同期の入出力を要求した際に、その要求を識別するために割り当てられる ID がリクエスト ID である。リクエスト ID を使用してデバイスへの要求完了待ち (tkse_wai_dev) を行うことができる。

リクエスト ID は 0 より大きい整数値である。

(6) データ番号 (INT 型)

デバイスからどのようなデータを読み込むかを、データ番号により指定する。データ番号は次の 2 種類に分類される。

・固有データ (データ番号≧0)

デバイス固有のデータで、データ番号はデバイスごとに定義される。

(例) ディスク　　データ番号＝物理ブロック番号
シリアル回線　　データ番号は 0 のみ使用

・属性データ (データ番号＜0)

デバイスの状態取得や設定、特殊機能などを指定する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.9.3 システムコール

デバイスのオープン

tkse_opn_dev

C言語インタフェース

```
ID  dd = tkse_opn_dev( UB *devnm, UINT omode ) ;
```

パラメータ

UB*	devnm	デバイス名
UNIT	o_mode	オープンモード
		(TD_READ ・ TD_WRITE ・ TD_UPDATE) |
		[TD_EXCL ・ TD_WEXCL ・ TD_REXCL]
		TD_READ　読み込み用オープン
		TD_WRITE　書き込み用オープン
		TD_UPDATE　更新(読み込み／書き込み)用オープン
		TD_EXCL　排他
		TD_WEXCL　排他書き込み
		TD_REXCL　排他読み込み

リターンパラメータ

ID	dd	＞ 0	デバイスディスクリプタ（正常終了）
		＜ 0	エラーコード

エラーコード

E_MACV	アドレス(dev,error)のアクセスは許されていない
E_BUSY	デバイス(dev)は既に排他的にオープンされている。
E_OACV	デバイス(dev)に対しての読み込みまたは書き込み操作は許されていない
E_NOEXS	デバイス(dev)は存在していない(登録されていない)
E_LIMIT	同時オープン可能な最大デバイス数を越えた
その他	デバイスドライバから返されたエラー

解説

dev で指定したデバイスを o_mode で指定したモードでオープンする。
オープンに成功するとデバイスディスクリプタを返す。
排他モード、および排他書込モードは、1 つのデバイスの同時オープンを制限する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスをオープンしたプロセスが終了した場合、自動的にそのデバイスはクローズされる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスのクローズ

tkse_cls_dev

C言語インタフェース

```
ER ercd = tkse_cls_dev( ID dd, UINT option );
```

パラメータ

ID	dd	デバイスディスクリプタ
UINT	option	クローズオプション TD_EJECT　メディア排出(排出不可能なデバイスの時は無視)

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	デバイスディスクリプタは存在していない
その他	デバイスドライバから返されたエラー

解説

dd で指定したデバイスをクローズする。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスのデータ読み込み（非同期）

tkse_rea_dev

C言語インタフェース

```
ID  reqid = tkse_rea_dev( ID dd, INT start, VP buf, INT size, TMO tmout );
```

パラメータ

ID	dd	デバイスディスクリプタ
INT	start	読み込み開始位置（≧0：固有データ、＜ 0：属性データ）
VP	buf	読み込みデータの格納領域
INT	size	読み込みデータサイズ
TMO	tmout	要求受付けタイムアウト時間（ミリ秒）

リターンパラメータ

ID	reqid	＞ 0	リクエスト ID（正常終了）
		＜ 0	エラーコード

エラーコード

E_MACV	アドレスのアクセスは許されていない
E_ID	デバイスディスクリプタは存在していない、または D_WRITE オープンである。
E_OACV	デバイス(dd)に対しての読み込み操作は許されていない
E_NOMDA	デバイス(dd)のメディアが存在しない
E_LIMIT	最大リクエスト数を越えた
E_TMOUT	タイムアウト
E_ABORT	処理中止
その他	デバイスドライバから返されたエラー

解説

dd で指定したデバイスから、start で指定した開始位置から size で指定したサイズのデータの読み込みを開始し、buf で指定した領域に格納する。

読み込みの完了を待たずに呼び出し元に戻る。読み込みが完了するまで buf の領域を保持しておかなくてはならない。正常に読み込みを開始した場合は場合は戻値としてリクエスト ID を返す。

size ＝ 0 の場合は実際の読み込みは行わず、現時点で読み込み可能なデータサイズを調べる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスのデータ読み込み（同期）

tkse_srea_dev

C言語インタフェース

```
ER  ercd = tkse_srea_dev( ID dd, INT start, VP buf, INT size, INT *asize );
```

パラメータ

ID	dd	デバイスディスクリプタ
INT	start	読み込み開始位置（≧0：固有データ、＜ 0：属性データ）
VP	buf	読み込みデータの格納領域
INT	size	読み込みデータサイズ
INT*	asize	読み込んだデータサイズ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_MACV	アドレスのアクセスは許されていない
E_ID	デバイスディスクリプタは存在していない、または D_WRITE オープンである。
E_OACV	デバイス(dd)に対しての読み込み操作は許されていない
E_NOMDA	デバイス(dd)のメディアが存在しない
E_LIMIT	最大リクエスト数を越えた
E_ABORT	処理中止
その他	デバイスドライバから返されたエラー

解説

dd で指定したデバイスから、start で指定した開始位置から size で指定したサイズのデータを読み込み、buf で指定した領域に格納する。

読み込みの完了を待って呼び出し元に戻る。読み込みに成功した場合は asize に実際に読み込んだサイズを返す。

size ＝ 0 の場合は実際の読み込みは行わず、現時点で読み込み可能なデータサイズを asize に返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスへのデータ書き込み（非同期）

tkse_wri_dev

C言語インタフェース

```
ID  reqid = tkse_wri_dev( ID dd, INT start, VP buf, INT size, TMO tmout );
```

パラメータ

ID	dd	デバイスディスクリプタ
INT	start	書き込み開始位置（≧0：固有データ、＜ 0：属性データ）
VP	buf	書き込みデータの格納領域
INT	size	書き込みデータサイズ
TMO	tmout	要求受付けタイムアウト時間（ミリ秒）

リターンパラメータ

ID	reqid	＞ 0	リクエスト ID（正常終了）
		＜ 0	エラーコード

エラーコード

E_MACV	アドレス(buf,a_size,error)のアクセスは許されていない
E_ID	デバイスディスクリプタは存在していない、または D_READ オープンである。
E_OACV	デバイス(dev)に対しての書き込み操作は許されていない
E_NOMDA	デバイス(dd)のメディアが存在しない
E_RONLY	デバイス(dev)は書込不可である
E_LIMIT	最大リクエスト数を越えた
E_TMOUT	タイムアウト
E_ABORT	処理中止
その他	デバイスドライバから返されたエラー

解説

buf で指定した領域のデータを、dd で指定したデバイスに、start で指定した開始位置から size で指定したサイズだけ書き込み開始する。

書き込みの完了を待たずに呼び出し元に戻る。書き込みが完了するまで buf の領域および内容を保持しておかなくてはならない。正常に書き込みを開始した場合は場合は戻値としてリクエスト ID を返す。

size ＝ 0 の場合は実際の書き込みは行わず、現時点で書き込み可能なデータサイズを調べる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスへのデータ書き込み（同期）

tkse_swri_dev

C言語インタフェース

```
ER  ercd = tkse_swri_dev( ID dd, INT start, VP buf, INT size, INT *asize );
```

パラメータ

ID	dd	デバイスディスクリプタ
INT	start	書き込み開始位置（≧0：固有データ、＜ 0：属性データ）
VP	buf	書き込みデータの格納領域
INT	size	書き込みデータサイズ
INT	asize	書き込んだデータサイズ

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(buf,a_size,error)のアクセスは許されていない
E_ID	デバイスディスクリプタは存在していない、または D_READ オープンである。
E_OACV	デバイス(dev)に対しての書き込み操作は許されていない
E_NOMDA	デバイス(dd)のメディアが存在しない
E_RONLY	デバイス(dev)は書込不可である
E_LIMIT	最大リクエスト数を越えた
E_ABORT	処理中止
その他	デバイスドライバから返されたエラー

解説

buf で指定した領域のデータを、dd で指定したデバイスに、start で指定した開始位置から size で指定したサイズだけ書き込む。

書き込みの完了を待って呼び出し元に戻る。書き込みに成功した場合は asize に実際に書き込んだサイズを返す。

size ＝ 0 の場合は実際の書き込みは行わず、現時点で書き込み可能なデータサイズを asize に返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスへの要求完了待ち

tkse_wai_dev

C言語インタフェース

```
ID  reqid = tkse_wai_dev( ID dd, ID reqid, INT *asize, ER *ioer, TMO tmout );
```

パラメータ

ID	dd	デバイスディスクリプタ
ID	reqid	リクエストID
INT*	asize	読み込み/書き込みサイズ
ER*	ioer	入出力エラー
TMO	tmout	タイムアウト時間（ミリ秒）

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_ID	ddまたはreqidが不正である
E_OBJ	reqidの要求は他のタスクで完了待ちしている
E_NOEXS	処理中の要求はない（reqid = 0の場合）
E_TMOUT	タイムアウト
E_ABORT	処理中止
その他	デバイスドライバから返されたエラー

解説

ddで指定したデバイスに対する要求reqidの完了を待つ。
reqid = 0 の場合は、ddに対する要求のうちのいずれかが完了するのを待つ。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスのサスペンド要求

tkse_sus_dev

C言語インタフェース

```
ER ercd = tkse_sus_dev(UINT mode);
```

パラメータ

UINT	mode	モード指定	
		D_EMRGSUS ・ D_SUSPEND | [D_FORCE]	
		・ D_DISSUS ・ D_ENASUS ・ D_CHECK	
		・ D_NOTIFY | [D_NOTSUS] | [D_NOTRES]	
		D_SUSPEND	サスペンドする
		D_DISSUS	サスペンドを禁止する
		D_ENASUS	サスペンドを許可する
		D_CHECK	サスペンド禁止カウントを調べる
		D_EMRGSUS	緊急サスペンドする
		D_FORCE	強制サスペンド指定
		D_NOTIFY	通知を要求する
		D_NOTSUS	サスペンド通知
		D_NOTRES	レジューム通知

リターンパラメータ

ER	ercd	≧ 0	正常終了 (D_CHECK のときはサスペンド禁止要求カウント数)
		＜ 0	エラーコード

エラーコード

E_BUSY	サスペンド禁止状態のためサスペンドできなかった
E_PAR	パラメータが不正である
E_LIMIT	サスペンド禁止要求カウントがシステムの制限を超えた
E_DISWAI	メッセージハンドラが起動されたため、処理が中断された

解説

モード指定 mode で指定した、システムのサスペンド関連制御を行う。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイス名の取得

tkse_get_dev

C言語インタフェース

```
ID  pdid = tkse_get_dev( ID devid, UB *devnm )
```

パラメータ

ID	devid	デバイス ID
UB*	devnm	デバイス名の格納領域

リターンパラメータ

ID	pdid	≧ 0	正常終了(物理デバイス ID)
		< 0	エラーコード

エラーコード

E_MACV	アドレス(dev)のアクセスは許されていない
E_NOEXS	デバイス ID は存在していない

解説

devno で指定したデバイス ID を持つデバイスのデバイス名を取り出し、devnm で指定した領域に格納する。
戻値として、デバイスが属する物理デバイスのデバイス ID を返す。

指定するデバイス ID は、デバイスイベントで得られたデバイス番号と同一である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスの情報の取得

tkse_ref_dev

C言語インタフェース

```
ID  devid = tkse_ref_dev( UB *devnm, T_RDEV *rdev );
```

パラメータ

UB*	devnm	対象デバイス名
T_RDEV*	rdev	デバイス管理情報の格納領域

リターンパラメータ

ID	devid	≧ 0	正常終了(デバイス ID)
		< 0	エラーコード

デバイス管理情報

```
typedef struct  {
    ATR     devatr;     /* デバイスの属性 */
    INT     blksz;      /* 物理ブロックサイズ(－１:不明) */
    INT     nsub;       /* サブユニット数 */
    INT     subno;      /* 0：物理デバイス、1～nsub：サブユニット番号+1 */
} T_RDEV;
```

エラーコード

E_MACV	アドレスのアクセスは許されていない
E_NOEXS	デバイスは存在していない

解説

devnm で指定したデバイスの情報を取得し、rdev で指定した領域に格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

デバイスの情報の取得

tkse_oref_dev

C言語インタフェース

```
ID  devid = tkse_oref_dev( ID dd, T_RDEV *rdev ) ;
```

パラメータ

ID	dd	対称デバイスディスクリプタ
T_RDEV*	rdev	デバイス管理情報の格納領域

リターンパラメータ

ID	devid	≧ 0	正常終了(デバイス ID)
		＜ 0	エラーコード

デバイス管理情報

```
typedef struct  {
    ATR     devatr;     /* デバイスの属性 */
    INT     blksz;      /* 物理ブロックサイズ(－１:不明) */
    INT     nsub;       /* サブユニット数 */
    INT     subno;      /* 0：物理デバイス、1～nsub：サブユニット番号+1 */
} T_RDEV;
```

エラーコード

E_MACV	アドレスのアクセスは許されていない
E_NOEXS	デバイスは存在していない

解説

dd で指定したデバイスの情報を取得し、rdev で指定した領域に格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

登録済みデバイスの取得

tkse_lst_dev

C言語インタフェース

```
INT  num = tkse_lst_dev( T_LDEV *ldev, INT start, INT ndev ) ;
```

パラメータ

T_LDEV	*ldev	登録デバイス情報の格納領域（配列）
INT	start	開始番号
INT	ndev	取得数

リターンパラメータ

INT	num	≧ 0	正常終了(残りの登録デバイス数)
		< 0	エラーコード

```
typedef struct {
        ATR     devatr;         /* デバイスの属性 */
        INT     blksz;          /* 固有データのブロックサイズ（-1：不明）
        INT     nsub;           /* サブユニット数 */
        UB      devnm[L_DEVNM]; /* 物理デバイス名 */
} T_LDEV;
```

エラーコード

E_MACV	アドレス(dev)のアクセスは許されていない
E_NOEXS	start が登録数を超えている
E_PAR	パラメータが不正である(ndev< 0)

解説

登録済みのデバイス情報を取得し、ldev で指定した領域に格納する。
戻値として残りのデバイス数を返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.10 時間管理

4.10.1 概要

Standard Extension の時間管理機能は、システム内部の基準時間であるシステム時間の取り出し／設定、およびカレンダー日付時刻との変換の機能を提供する。
システム時間は、ファイルの生成／更新／アクセス日時等のシステム内部の時間表現として使用される。

システム時間はグリニッジ標準時 GMT (Greenwich Mean Time) の 1985 年 1 月 1 日 00:00 からの秒数で表される 32 ビットの値であり、以下に示すように定義される。

```
typedef W   STIME;
```

システム時間に対して、マシンが実際に存在する地域の時間をローカル時間と呼ぶ。時間管理機能はシステム時間とローカル時間との時差を保持しているため、ローカル時間の日付時刻の取り出し／設定機能も提供している。

システム時間とローカル時間の関係は、以下に示す時間補正データとして定義される。

```
typedef struct {
        W       adjust;         /* システム時間との時差(秒) */
        W       dst_flg;        /* DST の適用タイプ */
        W       dst_adj;        /* DST 調整時間(分) */
} TIMEZONE;
```

dst_flg はサマータイム (Daylight Saving Time, DST) の適用タイプを示す。0 は適用しないことを示し、0 以外の値は適用することを示す。0 以外の値の意味は時間管理機能では関知せず、単に 0 か 0 でないかの判断のみを行う。

dst_adj は DST による時間の調整時間（分）を示す －(12×60) ～ +(12×60) の値である。時間管理機能は DST が適用されるか否かの判断は行わない。時間管理機能を使用するアプリケーションが、DST が実際に適用される期間の開始時に dst_adj を適当な値に設定し、DST が適用される期間の終了時に dst_adj を 0 に設定する必要がある。

時間補正データにより、ローカル時間は以下の式で定義される。

```
ローカル時間(秒)
        ＝ システム時間(秒) - adjust + (dst_flg ? (dst_adj×60): 0)
```


Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

時間管理機能はシステム時間だけではなく、以下に示す構造体で定義されるカレンダー日付時刻をサポートする。また、システム時間とカレンダー日付時刻との間の相互変換機能を提供する。

```
typedef struct {
        W       d_year;   /* 1900 年からのオフセット(85～) */
        W       d_month;  /* 月 ( 1 ～ 12, 0) */
        W       d_day;    /* 日 ( 1 ～ 31 ) */
        W       d_hour;   /* 時 ( 0 ～ 23 ) */
        W       d_min;    /* 分 ( 0 ～ 59 ) */
        W       d_sec;    /* 秒 ( 0 ～ 59 ) */
        W       d_week;   /* 週 ( 1 ～ 54 ) */
        W       d_wday;   /* 曜日 ( 0 ～ 6, 0が日曜) */
        W       d_days;   /* 日 ( 1 ～ 366 ) */
} DATE_TIM;
```

d_week は、その年の 1 月 1 日の週を 1 とした場合の週の数を意味し、d_days は、その年の 1 月 1 日を 1 とした場合の日数を意味する。また、d_month ＝ 0 は特殊な意味として使用される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.10.2 システムコール

システム時刻参照

tkse_get_tim

C言語インタフェース

```
ER ercd = tkse_get_tim(SYSTIM *pk_tim);
```

パラメータ

SYSTIM*	pk_tim	現在時刻を返すパケットアドレス

pk_timの内容

SYSTIM	systim	システム設定用の現在時刻

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である（pk_timが不正）

解説

システム時刻の現在の値を取得してpk_timに返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

システム時刻設定

tkse_set_tim

C言語インタフェース

```
ER ercd = tkse_set_tim(SYSTIM *pk_tim);
```

パラメータ

SYSTIM*	pk_tim	現在時刻を指定するパケットアドレス

pk_tim の内容

SYSTIM	systim	システム設定用の現在時刻

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である（pk_tim が不正）

解説

システム時刻の値を pk_tim で指定した値に設定する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

システム稼働時間参照

tkse_get_otm

C言語インタフェース

```
ER ercd = tkse_get_otm(SYSTIM *pk_tim);
```

パラメータ

SYSTEM*	pk_tim	稼働時間を返すパケットアドレス

pk_tim の内容

SYSTIM	opetim	システム設定用の現在時刻

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_PAR	パラメータが不正である（pk_tim が不正）

解説

システム稼働時間を取得する。

システム稼働時間はシステム時刻と異なり、システム起動時を 0 とする稼働時間である。

システム稼働時間は tkse_set_tim による時刻設定に影響されない。また、システム稼働時刻の精度はシステム時刻と同じである。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.10.3 ライブラリコール

システム時間、カレンダー日付、ローカル時間、グリニッチ標準時刻(GMT)をそれぞれ変換するためのライブラリ関数を定義する。

通算秒数として有効なのは 0x00000000-24*60*60 ～ 0x7fffffff+24*60*60 の範囲である(TIMEZONE による時間補正のため、システム時間の前後 1 日を含んでいる)。年数として有効なのは、1985 年～2053 年となる。
関数の引数としてカレンダー時刻を持つものに関して、不正なカレンダー時刻を指定した場合、戻値の内容は保証されない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

カレンダー日付から通算秒数への変換

DATEtoTIME

C言語インタフェース

```
void DATEtoTIME(STIME *time, DATE_TIM *date);
```

パラメータ

STIME	*time	通算秒数を格納する領域
DATE_TIM	*date	カレンダー時刻

リターンパラメータ

なし

解説

date で指定したカレンダー日付を、1985 年 1 月 1 日 00:00:00 からの通算秒数に変換して time に格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

通算秒数からカレンダー日付への変換

TIMEtoDATE

C言語インタフェース

```
void TIMEtoDATE(DATE_TIM *date, STIME time);
```

パラメータ

DATE_TIM	*date	カレンダー時刻を格納する領域
STIME	time	通算秒数

リターンパラメータ

なし

解説

time で指定した 1985 年 1 月 1 日 00:00:00 からの通算秒数を、カレンダー時刻に変換して date に格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

ローカル時間補正

GMTtoLT

C言語インタフェース

```
STIME ltim = GMTtoLT(STIME time, TIMEZONE *tz);
```

パラメータ

STIME	time	通算秒数
TIMEZONE	*tz	時間補正

リターンパラメータ

STIME	ltim	ローカル時間

解説

time で指定したシステム(GMT)時間に対し、tz で指定した時刻補正を行い、ローカル時間に変換して返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

標準時間補正

LTtoGMT

C言語インタフェース

```
STIME gtim = LTtoGMT(STIME time, TIMEZONE *tz);
```

パラメータ

STIME	time	通算秒数
TIMEZONE	*tz	時間補正

リターンパラメータ

STIME	gtim	システム(GMT)時間

解説

time で指定したローカル時間に対し、tz で指定した時刻補正を行い、システム(GMT)時間に変換して返す。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.11 システム管理

4.11.1 概要

Standard Extension のシステム管理機能は、システムプログラムをロード／アンロードする機能、およびシステム情報を取得する機能を提供する。

(1) システムプログラムのロード／アンロード

システムプログラムとは、システムメモリ上に配置される T-Kernel ベースのプログラムのことである。T-Kernel と同じ保護レベル（レベル 0）で動作し、T-Kernel API の実行やシステムメモリのアクセスを行うことができる。

主としてデバイスドライバやサブシステムを登録するために使用する。

(2) システム情報の取得

システム情報として、Standard Extension のバージョン情報を取得することができる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.11.2 システムコール

システムプログラムのロード

tkse_lod_spg

C 言語インタフェース

```
ER ercd = tkse_lod_spg(T_LSPG *pk_sysprg, TC *arg, VW info[N_SPG_INFO]);
```

パラメータ

T_LSPG	*pk_sysprg	システムプログラム情報
TC*	arg	システムプログラムのロード時に引数として渡す文字列
VW	info[N_SPG_INFO]	ローディング情報の格納領域(N_SPG_INFO＝2)

```
typedef struct {
        ATR     spgatr  システムプログラム属性
        VP      spghdr  ロードするシステムプログラムへのハンドル

        /* その他の実装依存情報 */

} T_LSPG;
```

spgatr はシステムプログラムの属性を表し、以下の指定を行う。

spgatr ：＝ (TMA_SEIO ・ TMA_LINK ・ TMA_PTR)

TMA_SEIO	システムプログラムへのハンドルは、標準入出力のファイルのパス名である
TMA_LINK	システムプログラムへのハンドルは、標準ファイルシステムのファイルへのリンクである
TMA_PTR	システムプログラムへのハンドルは、メモリ中に展開されたコードへのポインタである

リターンパラメータ

ER	ercd	≧ 0	正常終了(システムプログラム ID)
		＜ 0	エラーコード

エラーコード

E_FACV	ファイルのアクセス権(E)がない(TMA_SEIO, TMA_LINK 指定時)

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

E_MACV	アドレス(info, hdr(TMA_PTR 指定時))のアクセスは許されていない
E_BUSY	ファイルは既に排他的にオープンされているためにオープンできなかった
E_IO	入出力エラーが発生した
E_NOEXS	ファイルは存在していない
E_NOFS	ファイルの属するファイルシステムは接続されていない
E_NOMEM	メモリ領域が不足した(ロードするメモリ領域が不足)
E_REC	ファイルにプログラムレコードが存在しない。またはプログラムレコードの内容が異常である(TMA_LINK 指定時)

解説

プログラムコードをシステムプログラムとしてシステムメモリ空間にロードし、一意のシステムプログラム ID を割り当てる。

T_LSPG 構造体の spgatr はシステムプログラムの属性を表す。

TMA_SEIO 属性を指定した場合、指定したファイルの内容をプログラムコードとしてロードする。spghdr に対象とするファイルの標準入出力のパス名を文字列で指定する。

TMA_LINK 属性を指定した場合、指定した標準ファイルシステムのファイル内の最初の実行プログラムレコードの内容をシステムプログラムとしてロードする。spghdr に対象とする標準ファイルシステムのファイルへのリンク（LINK*）を指定する。

TMA_PTR 属性を指定した場合、メモリ上のプログラムコードをシステムプログラムとする。spghdr にメモリ上のプログラムコード領域の先頭アドレスを指定する。なお、メモリ上のプログラムコードの形式は実装依存とする。

ロードが成功した場合、info[0]にロードした先頭アドレス、info[1]にロードした終端アドレスを返す。

システムプログラムはメモリ上に単純にロードされる（マッピングされる)だけであり、シンボルアドレスのリロケーション等の処理は行われない。また、すでにロードされているシステムプログラムと同じものを指定した場合も、新たに別のメモリ空間を割り当ててロードが行われる。この場合はそれぞれ別個のシステムプログラム ID が割り当てられる。

システムプログラムをロードした領域は常に常駐化される。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

システムプログラムのアンロード

tkse_unl_spg

C言語インタフェース

```
ER ercd = tkse_unl_spg(W progid);
```

パラメータ

W	progid	システムプログラム ID

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_ID	指定したシステムプログラムは存在しない

解説

progid で指定したシステムプログラムをアンロードする。システムプログラムのためにメモリ上にマップされていた領域は、すべてマップ解除される。このときシステムプログラムが使用中であるか否かは関知しない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

バージョンの取得

tkse_get_ver

C言語インタフェース

```
ER ercd = tkse_get_ver(T_VER* version);
```

パラメータ

T_VER	*version	バージョン情報の格納領域

```
typedef struct {
        UH      maker;      /* メーカー */
        UH      id;         /* 形式番号 */
        UH      spver;      /* 仕様書バージョン */
        UH      prver;      /* 製品バージョン */
        UH      prno[4];    /* 製品管理情報 */
        UH      cpu;        /* CPU 情報 */
        UH      var;        /* バリエーション記述子 */
} T_VER;
```

リターンパラメータ

ER	ercd	エラーコード

エラーコード

E_OK	正常終了
E_MACV	アドレス(version)のアクセスは許されていない

解説

Standard Extensionのバージョン情報を取得してversionに格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.12 共有ライブラリ

4.12.1 概要

Standard Extension の共有ライブラリ機能は、実行時に動的にロードする、複数のプロセスから共有可能なプログラムコード（ライブラリ）の管理を行う。
共有ライブラリはファイルシステム上の共有ライブラリファイルとして存在する。共有ライブラリ機能を使用したプログラムを実行する際に、共有ライブラリのロードおよびシンボル解決を行う。

共有ライブラリのロードおよびシンボル解決は、ライブラリとして提供される関数コールにより行う。また共有ライブラリをシステム構築時に指定する共有ライブラリパスに置くことにより、自動的にロードおよびシンボル解決を行うことができる。
以下にそれぞれの方法の特徴を記す。

(1) プログラム中でライブラリ関数を用いて共有ライブラリを利用
- 共有ライブラリのロードおよびシンボル解決は、明示的にライブラリ関数をコールする
- 共有ライブラリファイルの置き場所は任意である

(2) プロセス生成時に自動的に共有ライブラリを利用
- 共有ライブラリのロードは、プロセス生成時に自動的に行われる
- シンボル解決は実行時に自動的に行われる
- 共有ライブラリファイルは、システム構築時に指定した共有ライブラリパス上に置く必要がある

共有ライブラリ機能は、コンパイラ、リンカなどの言語処理系の機能に依存する。共有ライブラリファイルを生成するには、言語処理系が位置独立コード(Position Independent Code)を生成できることが必須条件である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

4.12.2 ライブラリコール

共有ライブラリのオープン

dlopen

C言語インタフェース

```
void *handle = dlopen(const char *filename, int flag);
```

パラメータ

const char*	filename		共有ライブラリファイルのパス名
int	flag		シンボル解決の指定
	(RTLD_LAZY ・ RTLD_NOW) | [RTLD_GLOBAL]		
	RTLD_LAZY (0x01)		未確定シンボルを実行時に順次解決する。
	RTLD_NOW (0x02)		ローディング時にすべての未確定シンボルを解決する。
	RTLD_GLOBAL (0x100)		シンボルをグローバルとする。

リターンパラメータ

void *	handle	＞ 0	正常終了（共有ライブラリへのハンドル）
		＝ 0	エラー

解説

filenameで指定したパス名の共有ライブラリを、自プロセスのローカル空間にロードする。
パス名は、標準入出力機能のパス名の仕様に従う。
ロードに成功すると、戻値に共有ライブラリへのハンドル(＞ 0) を返す。失敗した場合は NULL(0)を返す。filename ＝ NULLの場合はロードを行わず、メインプログラムのハンドルを返す。
得られたハンドルはdlsym()の引数に使用できる。

シンボル解決の指定はflagにRTLD_LAZYかRTLD_NOWのいずれかを指定する。また同時に、RTLD_GLOBALを論理和で指定することもできる。
RTLD_NOWを指定した場合､ライブラリ中の未定義シンボルをすべて解決した後にdlopen()は呼び出し元に戻る。解決できないシンボルが存在する場合はエラーを返す。
RTLD_LAZY を指定した場合、シンボルは実行時に必要になった時点で解決が行われる。実行時にシンボルを解決できなかった場合の動作は保証されない（通常は例外が発生する)。
RTLD_GLOBAL を指定した場合、ロードした共有ライブラリの外部シンボルが、後からオープンされた他の共有ライブラリのシンボル解決で利用できるようになる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

共有ライブラリのシンボル検索

dlsym

C言語インタフェース

```
void *val = dlsym(void *handle, const char *symbol);
```

パラメータ

void *	handle	共有ライブラリのハンドル
const char *	symbol	指定シンボル

リターンパラメータ

void *	val	≠ NULL	正常終了（シンボルの値）
		= NULL	エラー

解説

handle で指定した共有ライブラリから symbol で指定したシンボルを検索し、その値を返す。
シンボルが見つからない場合は NULL を返す。

特殊なハンドルとして、handle に RTLD_NEXT と RTLD_DEFAULT を指定できる。
handle に RTLD_NEXT を指定した場合、シンボル検索は dlsym()を呼び出した共有ライブラリの後にある「次の」共有ライブラリから始まる。
handle に RTLD_DEFAULT を指定した場合、シンボル検索は dlsym()を呼び出した共有ライブラリのスコープ内で行われる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

共有ライブラリのクローズ

dlclose

C言語インタフェース

```
int rtn = dlclose( void *handle );
```

パラメータ

void *	handle	共有ライブラリへのハンドル

リターンパラメータ

int	rtn	= 0	正常終了
		< 0	エラー

解説

handle で指定した共有ライブラリをクローズする。
同一のライブラリを複数回 dlopen() した場合は、オープンした回数だけ dlclose() した時にクローズされる。
クローズされた共有ライブラリに含まれるシンボルは、以降使用できない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

共有ライブラリのシンボル情報の取得

dladdr

C言語インタフェース

```
int rtn = dladdr( void *addr, Dl_info *info );
```

パラメータ

void *	addr	シンボルのアドレス
Dl_info *	info	指定したシンボルの情報

```
typedef struct {
        const char    *dli_fname;  /* ファイル名 */
        void          *dli_fbase;  /* ベースアドレス */
        const char    *dli_sname;  /* シンボル名 */
        void          *dli_saddr;  /* シンボルのアドレス */
} Dl_info;
```

リターンパラメータ

int	rtn	≠ 0	正常終了
		= 0	エラー

解説

addr で指定したアドレス値として持つシンボルが、現在オープンしている共有ライブラリに存在する場合、そのシンボル情報を info に返す。

dli_fname は共有ライブラリのファイル名へのポインタを格納する。ファイル名は標準入出力管理の仕様に準ずる。
dli_fbase は共有ライブラリのベースアドレス（ロードオフセット）を格納する。これは共有ライブラリのハンドルと等しい。
dli_sname は addr で指定した値と同じ、または小さい値で最も値が近いシンボルの名前へのポインタを格納する。
dli_saddr は dli_sname のシンボルの値（アドレス）を格納する。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

第5章　実装方式

5.1　概要

Standard Extension は T-Kernel の機能拡張プログラムであり、その機能は T-Kernel サブシステムとして実装される。Standard Extension が使用するサブシステムを以下に示す。

・メモリ管理サブシステム
・セグメント管理サブシステム
・プロセス／タスク管理サブシステム
・プロセス間メッセージサブシステム
・グローバル名サブシステム
・タスク間同期・通信サブシステム
・標準入出力サブシステム
・標準ファイル管理サブシステム
・イベント管理サブシステム
・デバイス管理サブシステム
・時間管理サブシステム

アプリケーションは Standard Extension の機能を呼び出すために、サブシステムの拡張 SVC として実装されたシステムコール（tkse_xxx_yyy）を利用する。このシステムコールは通常、アプリケーションにリンクされたインタフェースライブラリ経由で呼び出される。

5.2　メモリ管理とセグメント管理

Standard Extension のメモリ空間の管理は、メモリ管理サブシステムとセグメント管理サブシステムの２つのサブシステムで実現される。

メモリ管理サブシステムはブロック単位のメモリ領域の管理を行う。

メモリ領域を割り当てると、その領域を構成する個々のメモリブロックはページテーブルに登録される。ページテーブルは論理アドレスと物理アドレスの関連付け、およびメモリブロックの様々な属性を保持するためのデータ構造である。MMU はページテーブルの情報を用いて、論理―物理アドレス変換や、メモリ領域のアクセス制限を実現する。ページテーブルはプロセス毎に独立しており、プロセス生成時に初期化され、プロセス終了時に破棄される。

セグメント管理サブシステムは、メモリ空間をディスクにマップする機能や常駐・非常駐の属性設定など、仮想記憶とメモリ空間の管理を行う。

仮想記憶が有効になっている状態で物理メモリが不足すると、セグメント管理は現在使用していないメモリページをページファイルに書き出してメモリページを破棄し(ページアウト)、新たなメモリ領域として割り当てる。また、ページアウトしているメモリページに対するメモリアクセスが発生した場合は、そのメモリページをメモ

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

リ上に読み込み（ページイン）、アクセスを実行する。

ページイン、ページアウトの処理はタスクのコンテキストで実行される。このためタスク独立部を実行中に、物理メモリの不足が発生する可能性のある処理や、またはページアウト状態のメモリ領域へアクセスする処理を行ってはならない。タスク独立部として動作するコード、および動作中に参照するデータはすべて常駐メモリ上に配置する必要がある。

5.3 プロセス／タスク管理

(1) メモリ空間

プロセスのローカルメモリ空間はT-Kernelのタスク固有空間として実装される。すべてのプロセスは独立したタスク固有空間を持ち、プロセス内のタスクは、そのプロセスのタスク固有空間に所属する。

(2) リソースグループ

プロセスはそれぞれ独立したリソースグループを持つ。プロセス生成時に新たなリソースグループが生成され、プロセス終了時に削除される。

リソースグループは、プロセス管理情報やファイルディスクリプタ、カレントディレクトリ情報など、それぞれのプロセスに固有の情報を保持するために使用される。

(3) タスクの保護レベル

ユーザプロセス内のタスクは保護レベル３、システムプロセス内のタスクは保護レベル１で実行される。これらのプロセスから、ドライバやOSで使用するシステム領域（保護レベル０の領域）にアクセスすることはできない。

例外として、初期プロセスは保護レベル０で動作する。このため初期プロセスからシステム領域へのアクセスは可能である。

(4) タスク優先度の割り当てとスケジューリング

Standard Extensionのタスク優先度(sepri)は、T-Kernelタスク優先度(kpri)に以下のように割り当てられる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

[表 4] タスク優先度の割り当て

Standard Extension タスク優先度(sepri)	T-Kernel タスク 優先度(kpri)	優先度の割り当て方法	スライスタイム （ミリ秒）
－	1～7	未割り当て	－
0～127 （絶対優先度グループ）	8～135	kpri = sepri + 8	10（固定値）
－	136～137	未割り当て	－
128～191 （ラウンドロビングループ１）	138	kpri = 138（固定値）	192 - sepri
192～255 （ラウンドロビングループ２）	139	kpri = 139（固定値）	256 - sepri
－	140	未割り当て	－

絶対優先度グループのタスクは、Standard Extension タスク優先度にオフセット値(+8)を加えた値を T-Kernel タスク優先度に割り当てられる。優先度の大小関係が保持されるため、絶対優先度グループのタスクは T-Kernel と同じ優先度順のスケジューリングとなる。ただし、同一優先度のタスクが複数存在する場合は、Standard Extension では一定時間(10msec)ごとにラウンドロビン方式で平等にスケジューリングするのに対し、T-Kernel ではタスクのスライスタイムを明示的に設定しない限り、自動的に実行権が移ることはない。

ラウンドロビングループ１のタスクは、Standard Extension タスク優先度に関わらずすべて T-Kernel タスク優先度 138 に割り当てられる。ラウンドロビンのアルゴリズムは、タスクのスライスタイムを Standard Extension タスク優先度に応じて設定し、同一の T-Kernel タスク優先度内で優先順位を回転させることにより実現する。タスクのスライスタイムは、192 から Standard Extension タスク優先度を減算した値となる。

同様に、ラウンドロビングループ２のタスクはすべて T-Kernel タスク優先度 139 に割り当てる。ラウンドロビンの実現方式はラウンドロビングループ１と同じである。タスクのスライスタイムは、256 から Standard Extension タスク優先度を減算した値となる。

Standard Extension タスク優先度として割り当てられていない T-Kernel タスク優先度(1～7, 136～137, 140)は、Standard Extension から利用することはできない。T-Kernel アプリケーションはこれらの優先度を自由に利用可能である。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

5.4 プロセス間メッセージ

送信したメッセージの内容は、送信元が確保したシステムメモリ空間の領域にコピーされ、送信先プロセスのメッセージキューに挿入される。挿入されたデータは、メッセージ受信時に指定したバッファ領域に再度コピーされる。

メッセージ送受信時の同期はタスクイベントにより実現する。メッセージ送信時に CONFM 指定でメッセージ受信待ちを行う場合は、タスクイベント番号 1 を使用する。また tkse_brk_msg()による待ち解除を行うために、タスクイベント番号 2 を使用する。メッセージ管理以外のソフトウェアからプロセス内のタスクに、これらのタスクイベント番号を送信してはならない。

メッセージハンドラを使用する場合は、メッセージハンドラを登録したプロセスのメインタスクにタスク例外を発生させることで、メッセージの非同期受信を実現する。メッセージ管理以外のソフトウェアからプロセスのメインタスクにタスク例外を発生させてはならない。初期プロセスは保護レベル 0 で動作するため、T-Kernel の制限上タスク例外を発生させることができない。このためメッセージハンドラを利用して非同期にメッセージ受信を行うことはできない。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

5.5 タスク間同期・通信

タスク間同期・通信機能は、T-Kernel の該当するシステムコールを間接的に呼び出すことで機能を実現する。

Standard Extension の拡張機能として、プロセス終了時のオブジェクト自動削除（TA_DELEXIT 指定）がある。この機能を実現するため、各プロセスは生成したオブジェクトの一覧情報をリソースグループとして保持する。また、この一覧情報を各オブジェクトと関連付けるため、オブジェクトの拡張情報 exinf を使用する。このため Standard Extension タスク間同期・通信機能で拡張情報 exinf を使用することはできない。

5.6 デバイス管理

デバイス管理機能は、T-Kernel/SM のデバイス管理機能を間接的に呼び出すことで機能を実現する。ただし Standard Extension 上からデバイスの登録・抹消を行うことはできない。

5.7 時間管理

時間管理機能は、T-Kernel 時間管理機能を利用して実装されている。

システム時刻は、Standard Extension 起動時に時計ドライバで RTC 時刻を読み込み初期設定される。時計ドライバが存在しない、あるいは RTC 時刻が不正な場合、システム時刻の内容は不定となる。

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

第6章　コンフィグレーション

6.1　システム構成情報

T-Kernel と同様に、システム構成情報を変更することで Standard Extension のコンフィグレーションを行うことができる。

以下の定義情報を Standard Extension の標準として定める。

ただし N:は数値列情報を、S:は文字列情報を表す。

・製品情報

N: OS-Ver	製品バージョン 製品管理情報（最大４個）

tkse_get_ver で取得するバージョン情報の、製品バージョンと製品管理情報を設定する。
指定しない場合は 0 となる。

・プロセス／タスク管理

N: MaxProc	最大プロセス数
N: MaxSubTsks	最大プロセス内タスク数
N: SysStkSz	プロセス内タスクのシステムスタックサイズ（バイト数）
N: UsrStkSz	プロセス内タスクのデフォルトユーザスタックサイズ（バイト数）
N: MaxSysPrg	最大システムプログラム数

・セグメント管理

N: MaxMapID	最大ディスクマップ数
N: MaxDiskID	最大ディスク接続数
N: MaxPageIO	ディスク入出力時の最大連続ページ数
N: SyncPeriod	ディスクの同期間隔（ミリ秒）

設定した周期毎にディスクバッファの内容をディスクと同期する。

N: SafetyMargin	メモリ割り当て時に安全マージンとして残すページ数

・メモリ管理

N: SRsvMem	システム用メモリの最低確保サイズ（ページ数）

・プロセス間メッセージ

N: TotalMsgMax	最大メッセージサイズ（全メッセージの合計）
N: MaxMsgSz	最大メッセージサイズ（個々のメッセージの最大値）

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.

・タスク間同期・通信

N: TcBufLim　　タスク間同期・通信用バッファサイズ上限（バイト数）

・グローバル名

N: GlobalNameLimit　　グローバル名最大数

・時間管理

N: CmClkUpd　　時計更新通知（0:無効／1:有効）

時計更新の通知を毎分０秒に行う。通知は tkse_brk_msg を呼び出すことで行われる。

・標準ファイル管理

N: FmTskPri　　ファイル管理タスク優先度（共通タスク）

ファイル管理タスク優先度（各ファイルシステムタスク）

N: MaxOpenF　　同時オープン可能な最大ファイル数

N: SyncTimeOut　　同期タイムアウト時間（ミリ秒）

N: FmTimeStamp　　タイムスタンプ更新制御（0:更新あり／1:更新なし）

レコード読み込み時にタイムスタンプを更新するかどうかを設定する。

0 ならば更新を行う。1 ならば更新を行わない。

・イベント管理

N: EmTskPri　　デバイスイベント受信タスク優先度

N: MaxEvtQ　　イベントキューサイズ（バイト数）

N: MaxEvMsg　　イベントメッセージ要求の最大登録数

N: EvtLife　　イベント自動消滅時間（ミリ秒）

ホイールイベントの自動消滅時間を設定する。

・標準入出力

N: UxMaxOpenF　　同時オープン可能な最大ファイル数（プロセス単位）

N: UxFsTskPri　　ファイルシステムタスク優先度

N: UxSyncTimeOut　　同期タイムアウト時間（ミリ秒）

Copyright © 2006-2008 by T-Engine Forum. All rights reserved.