μITRONの基礎と実践プログラミング

μTRONの基礎

目次

■ 組込みOS概要

- ○ OSの役割
- ○ OSのメリットHデメリット
- ○ 組込みOSの種類および用途

■ リアルタイムOS概要

- ○ リアルタイムOSとは
- ○ リアルタイムOSのメモリ管理、スケジューリング方法

■ μITRON概要

- ○ μITRONの基本構造
- ○ μITRONの状態遷移

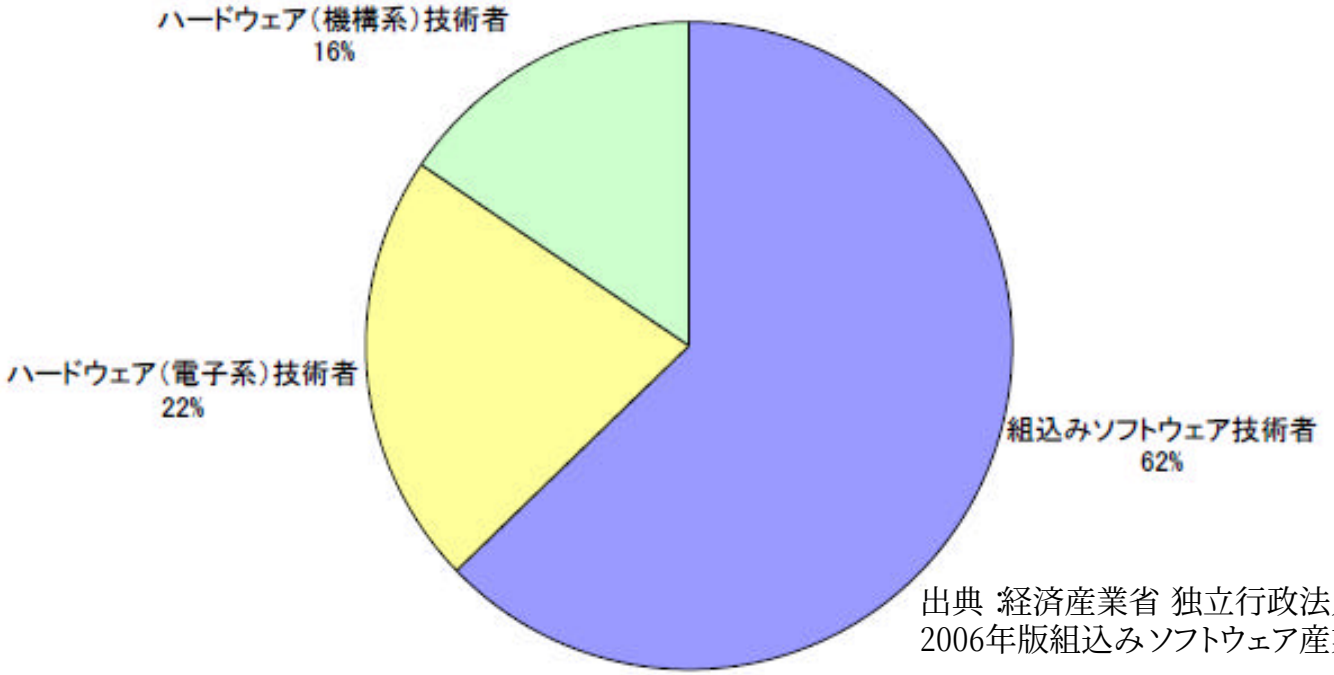
■ TOPPERS/JSPカーネル概要

- ○ TOPPERS/JSPカーネルのデータタイプ
- ○ TOPPERS/JSPカーネルの機能概要
- ○ TOPPERS/JSPカーネルによるプログラミング概要

ソフトウェア開発規模

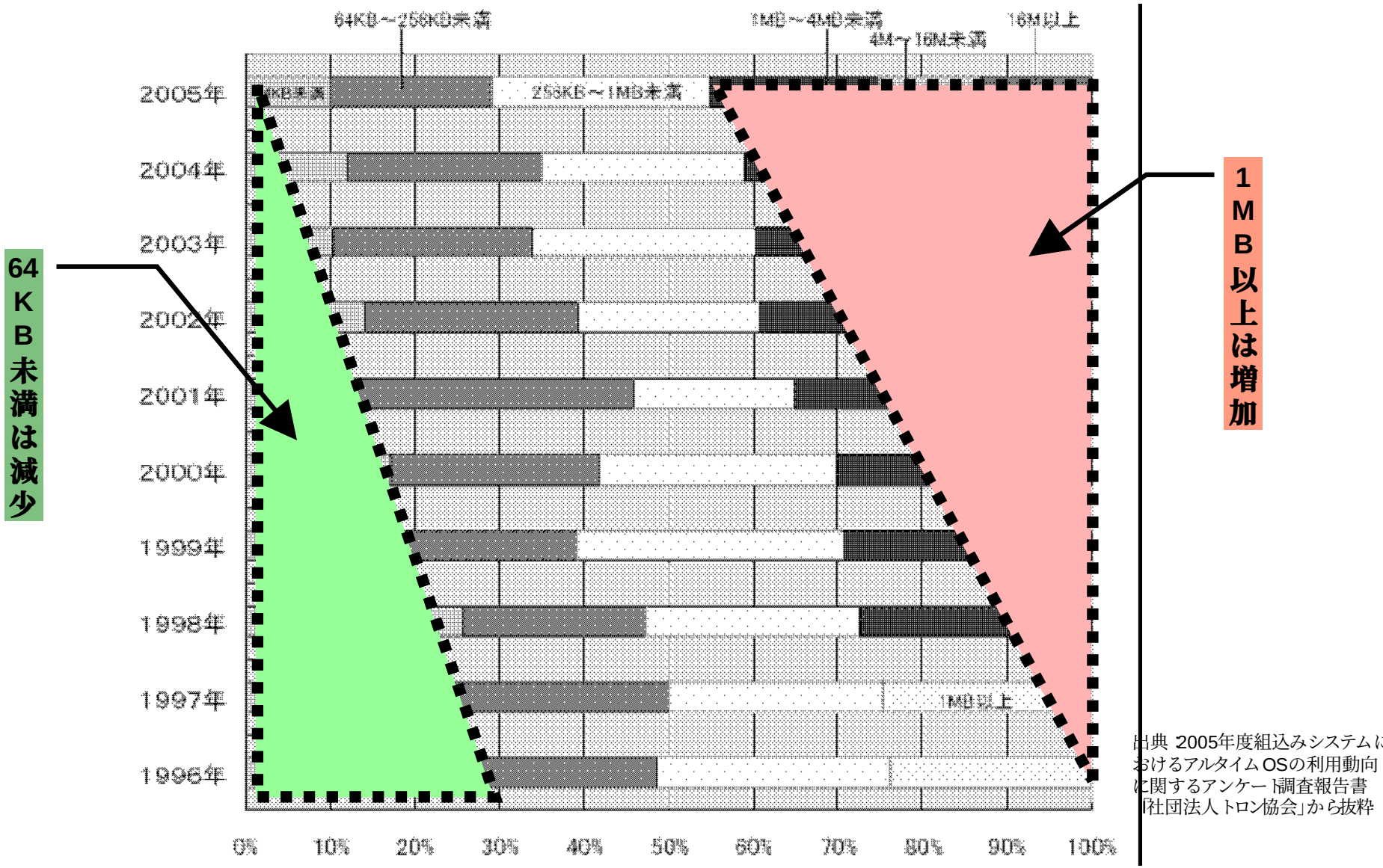
■推定組込みソフトウェア技術者数(2006年度調査)

- ●推定組込み技術者 :約31万1,000人
- ●推定ソフトウェア技術者 :約19万3,000人
  - ・2005年度 :約17万5,000人
  - ・2004年度 :約15万人



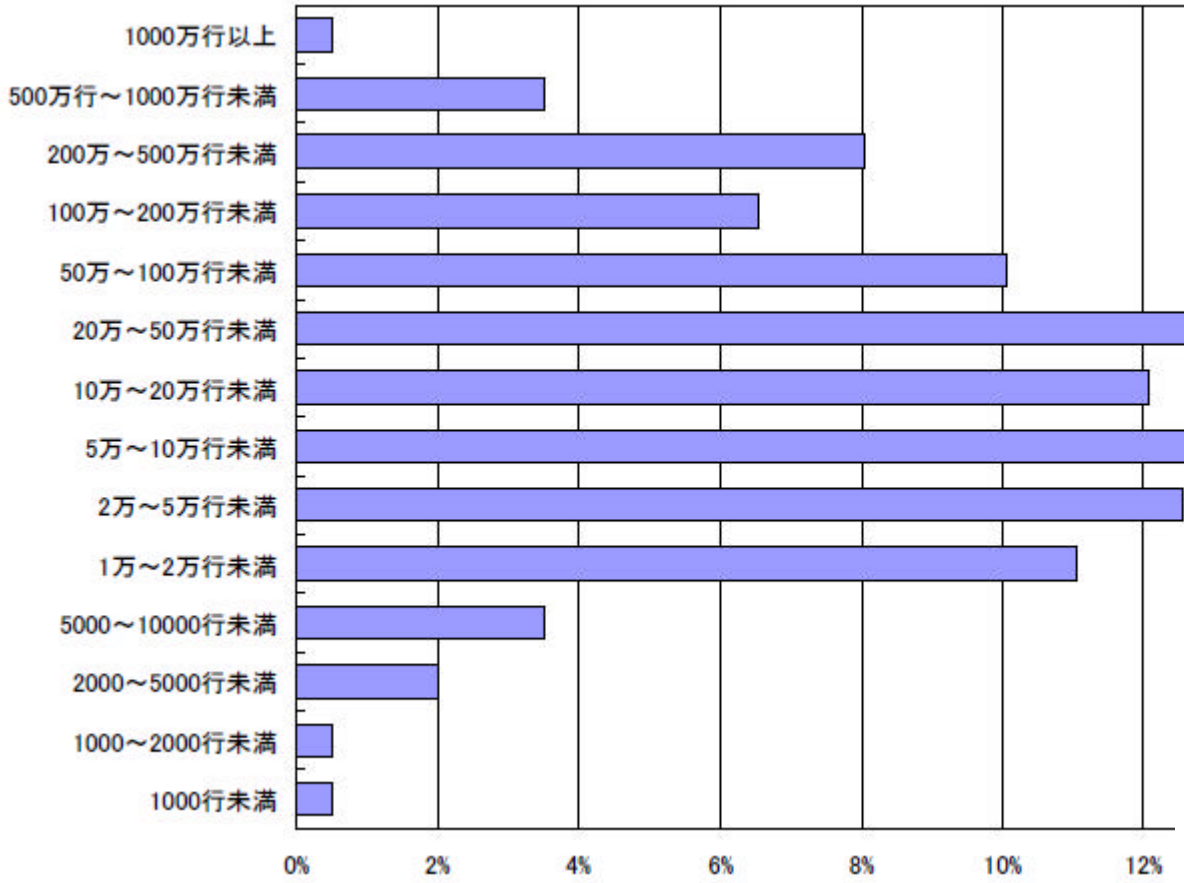
出典 :経済産業省 独立行政法人 情報処理推進機構
2006年版組込みソフトウェア産業実態調査 報告書

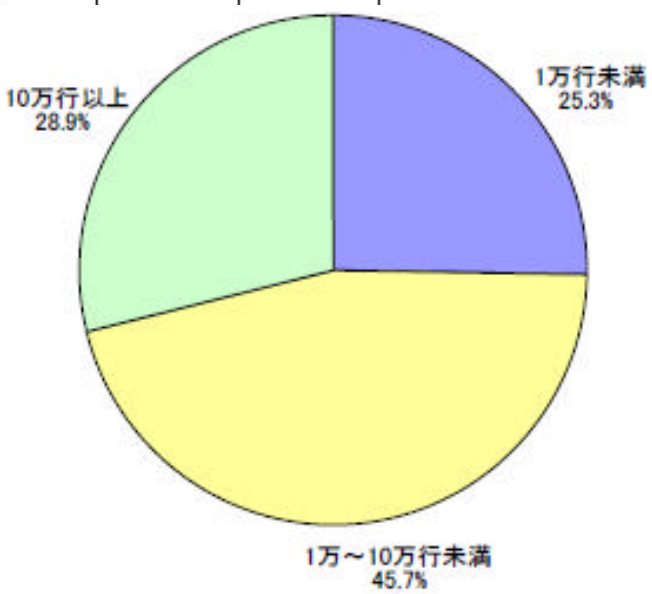
プログラムサイズの増大



出典 2005年度組込みシステムにおけるアルタイムOSの利用動向に関するアンケート調査報告書「社団法人トロン協会」から抜粋

プログラムステップ数





出典 :経済産業省 独立行政法人 情報処理推進機構
2006年版組込みソフトウェア産業実態調査　報告書

ソフトウェアの大規模化

■市場の要求による多機能化

- カラー液晶、カメラ、動画
- テレビ電話、電子マネー、地上デジタル放送

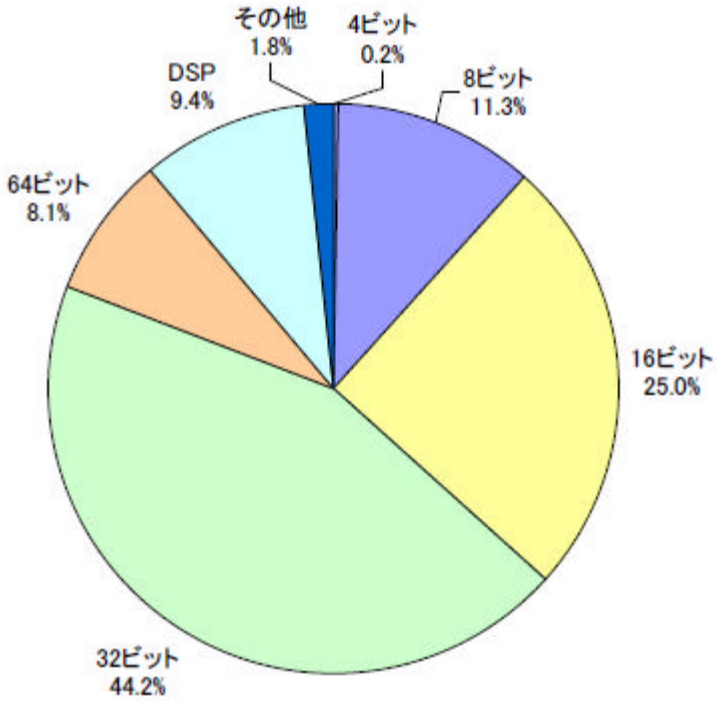
■開発規模・開発者の増大

- 500万行/1プロジェクト(平均310万行：新規開発行数)
- 400人/1プロジェクト

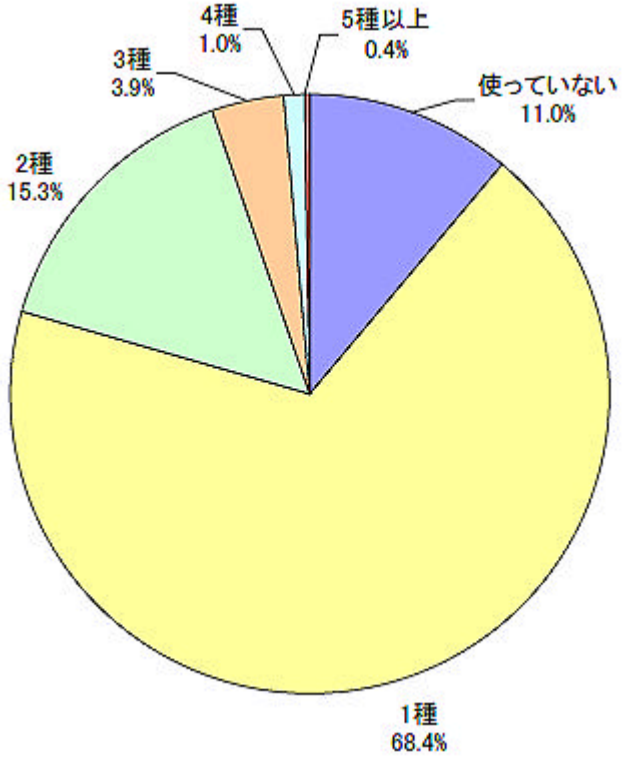
出典：経済産業省
独立行政法人 情報処理推進機構
2005年版組込みソフトウェア産業実態調査　報告書

市場の要求に応えるために :1

■高機能CPUの採用



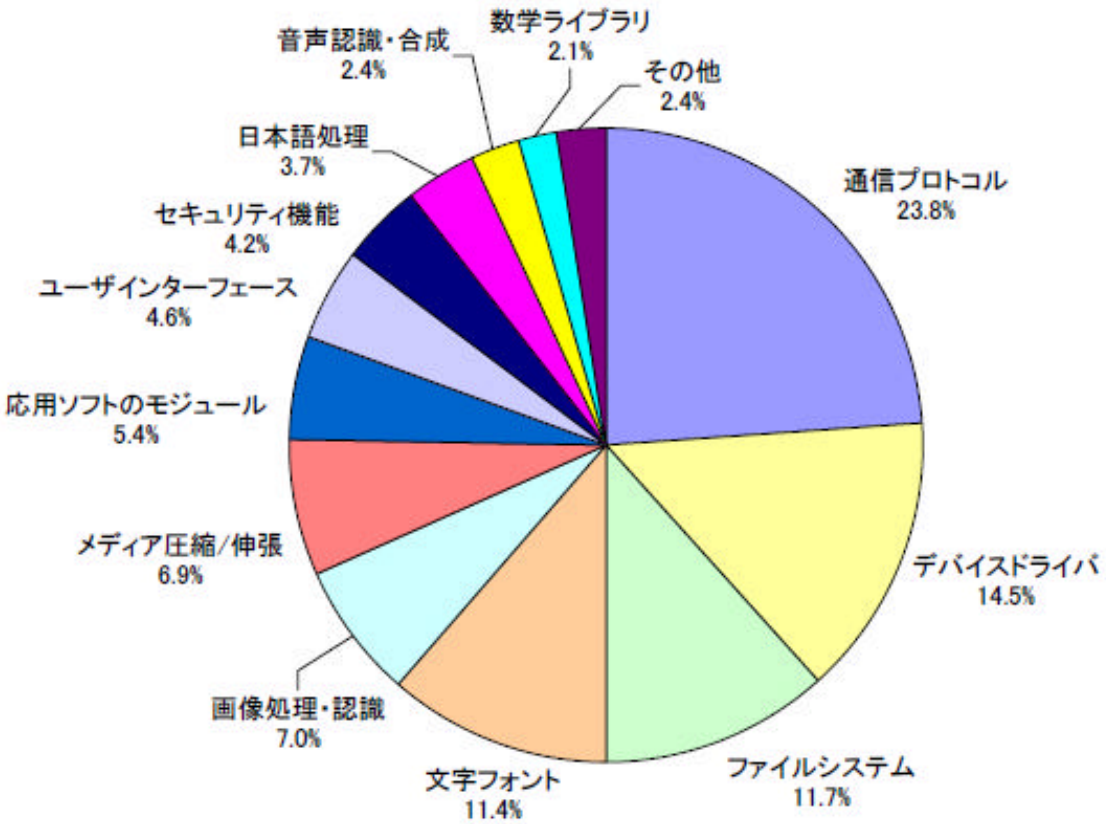
■負荷分散



出典 :経済産業省 独立行政法人 情報処理推進機構
2006年版組込みソフトウェア産業実態調査　報告書

市場の要求に応えるために 2

■ソフトウェア部品(市販ソフト)の活用
■ソフトウェア部品の再利用



出典 :経済産業省 独立行政法人 情報処理推進機構
2006年版組込みソフトウェア産業実態調査　報告書

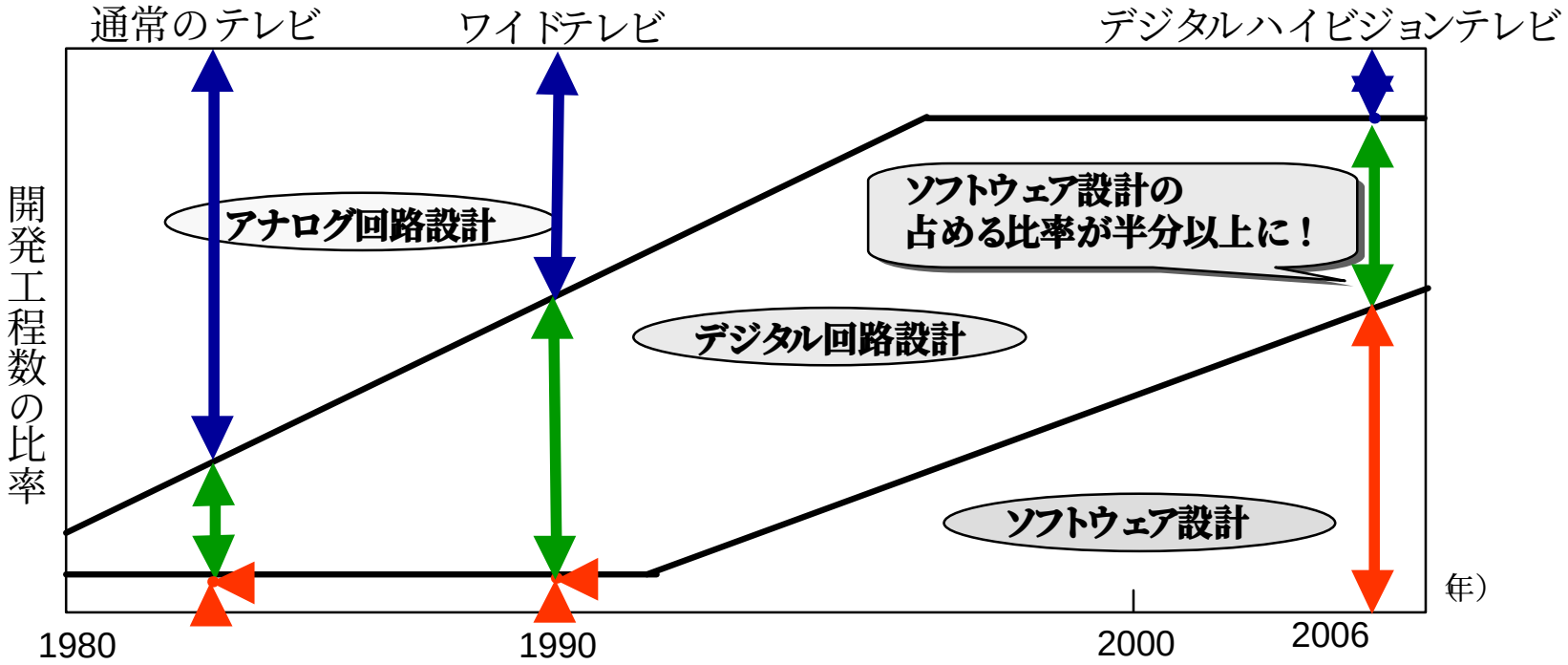
OSの必然性

■開発の比重：ハードウェア→ソフトウェアへ移行

例（テレビ受像器）	プログラム容量	人員
10年前：	**8Kバイト**	**1**
現行TV:	**50Kバイト**	**8～36**
ハイビジョンTV：	**1Mバイト**	**40～400**
インターネットTV：	**4Mバイト**	**220～16000**

プログラムがN倍増えると開発工数：N～N^2増加

■増大するソフトウェア技術者を押さえるためにOSを使用



OSの役割

コンピュータシステム

- ハードウェア :CPU、メモリ、I/O
- ソフトウェア :アプリケーションソフトウェア、ミドルウェア、デバイスドライバ、OS

OSとは

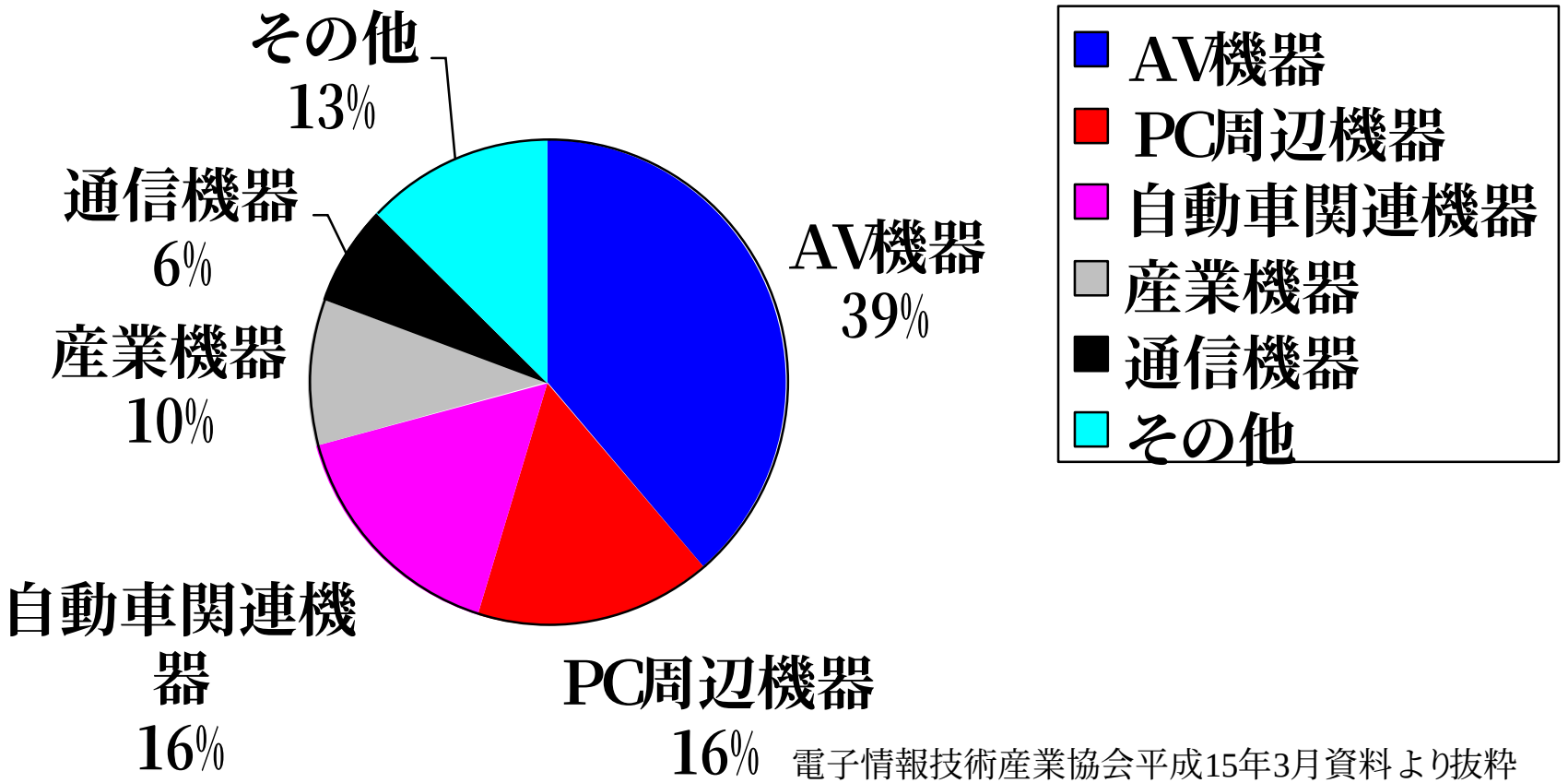
- ハードウェアを制御するための基本ソフトウェア
- コンピュータシステムを運用するためのソフトウェア

組込み用OS

- ハードウェア(CPU、周辺デバイス)の隠蔽
- リアルタイム処理
- マルチプログラミング環境の提供
- ミドルウェア/ソフトウェア部品の共通化
- ・・・・

OSを使用している製品分野

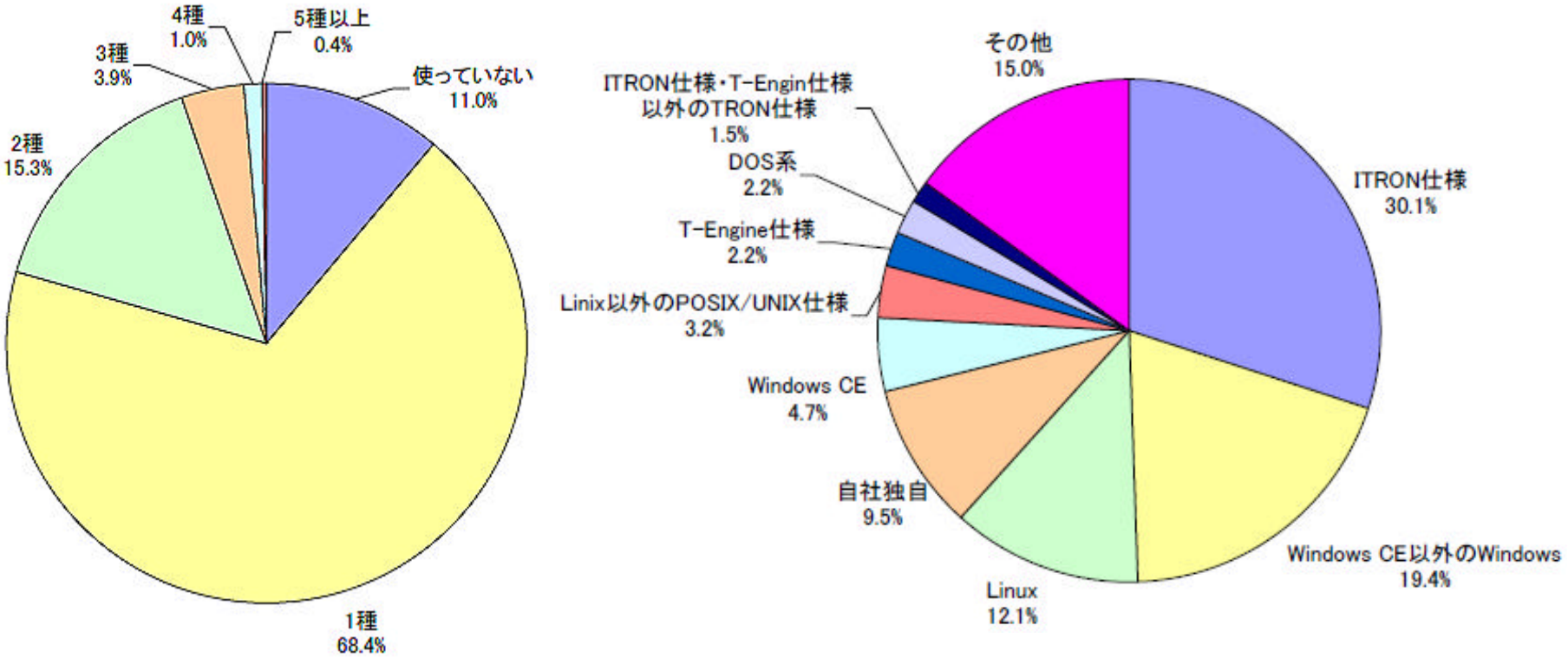
有効回答数 :31



電子情報技術産業協会平成15年3月資料より抜粋

使用しているOSの種類

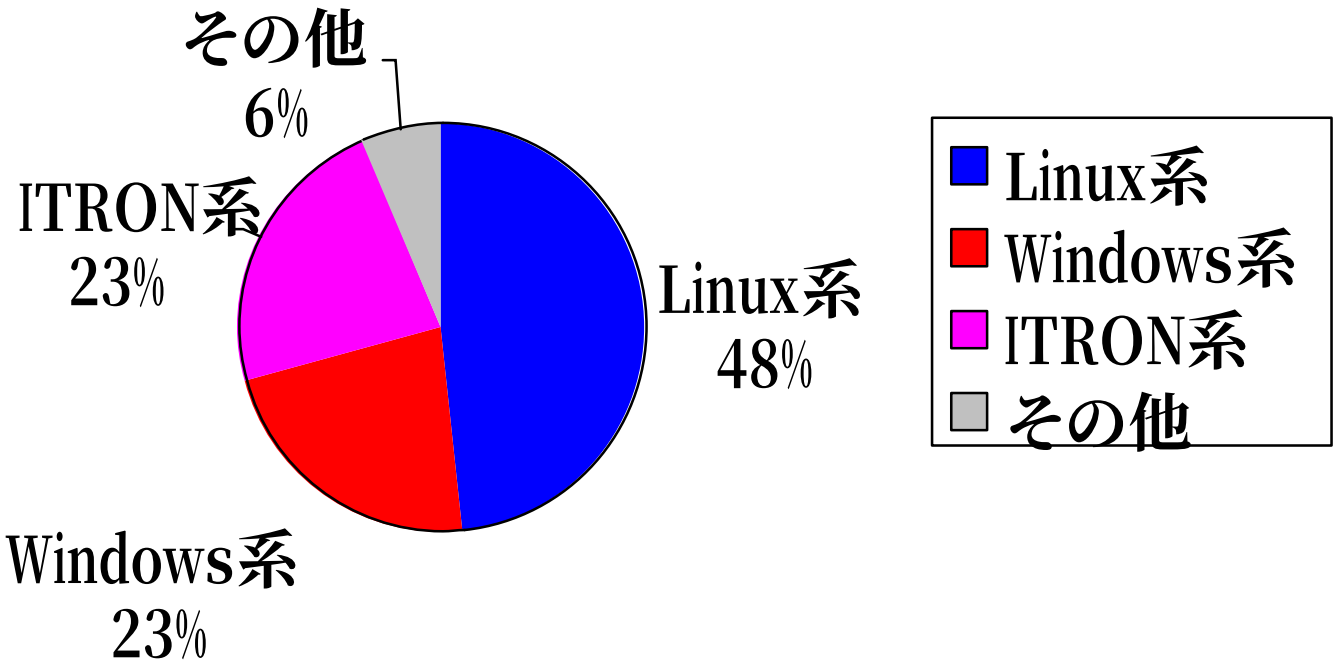
■ターゲットOSの使用種類



出典 :経済産業省 独立行政法人 情報処理推進機構
2006年版組込みソフトウェア産業実態調査　報告書

これから市場で多く使われるOSは何だと思われますか。

複数回答扱い
設問に対する回答人数：29



電子情報技術産業協会平成15年3月資料より抜粋

これから市場で多く使われるOSは何だと思われますか。理由

■Linux系

- ライセンスフリー（15件中5件）
- ソースコード公開

■Windows系

- 普及率が高い
- デファクトスタンダードになっている。

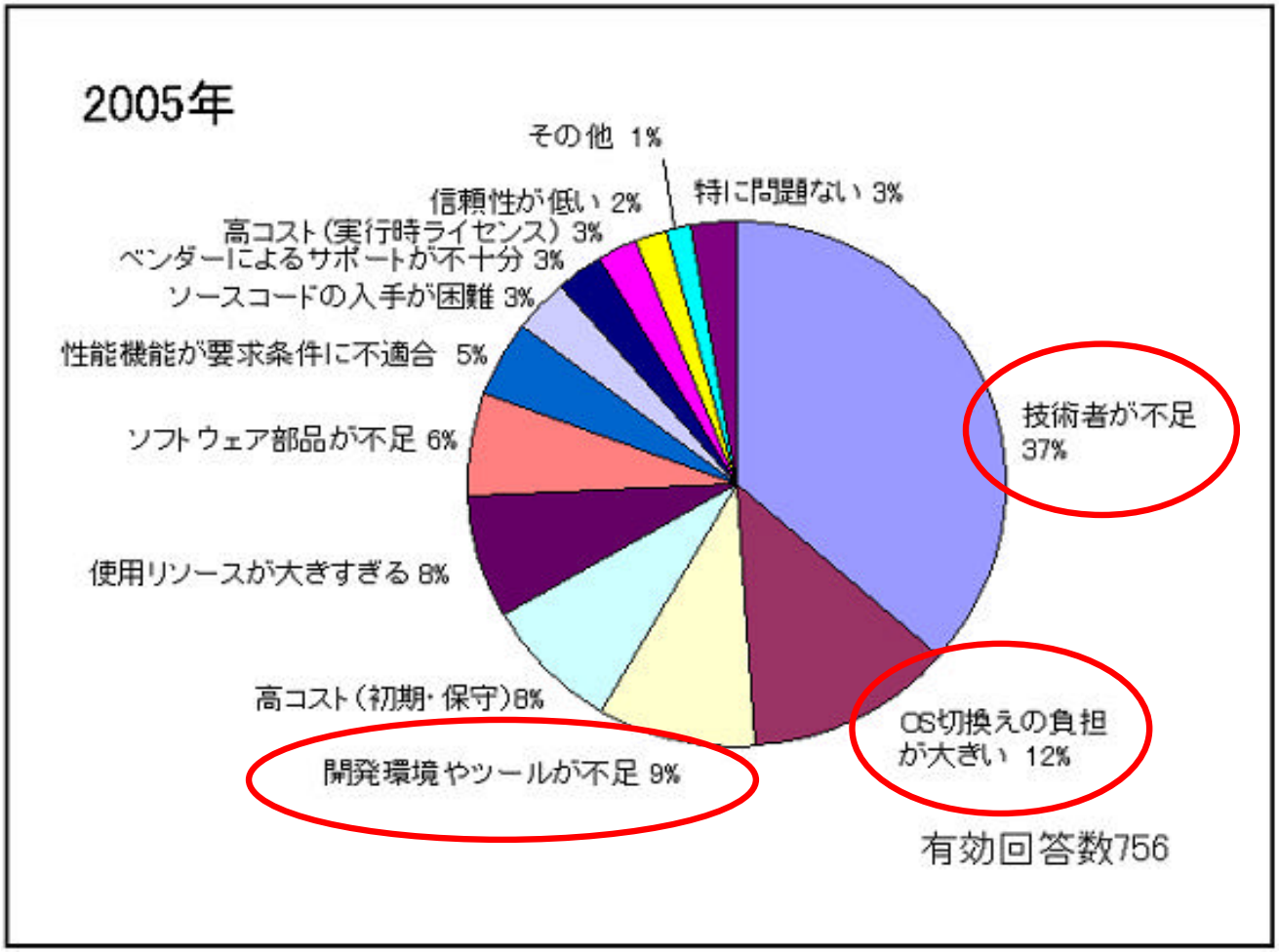
■ITRON

- オープンアーキテクチャー
- 小さい、安い、早い

複数回答扱い
設問に対する回答人数：29

電子情報技術産業協会平成15年3月資料より抜粋

使用しているOSの不満な点



リアルタイムOSの問題点

2005年度
組込みシステムにおけるリアルタイムOS の利用動向に関するアンケート調査報告書
「TRON協会」から抜粋

OSのメリット

■設計段階

- ●モジュール化が容易。
- ●ソフトウェア開発者がアプリケーションだけに専念できる。
- ●タスク単位（最小モジュール単位）で思考できる。
- ●タスク間通信方法だけ決めれば、複数人で各タスクを独立して設計可能。

■デバッグ時

- ●最小単位であるタスク毎にデバッグが行える。
- ●タスク毎のデバッグが可能なため、複数人数で同時にデバッグが可能。
- ●最終的な統合デバッグでは、タスク間通信や同期をテストするだけで済む

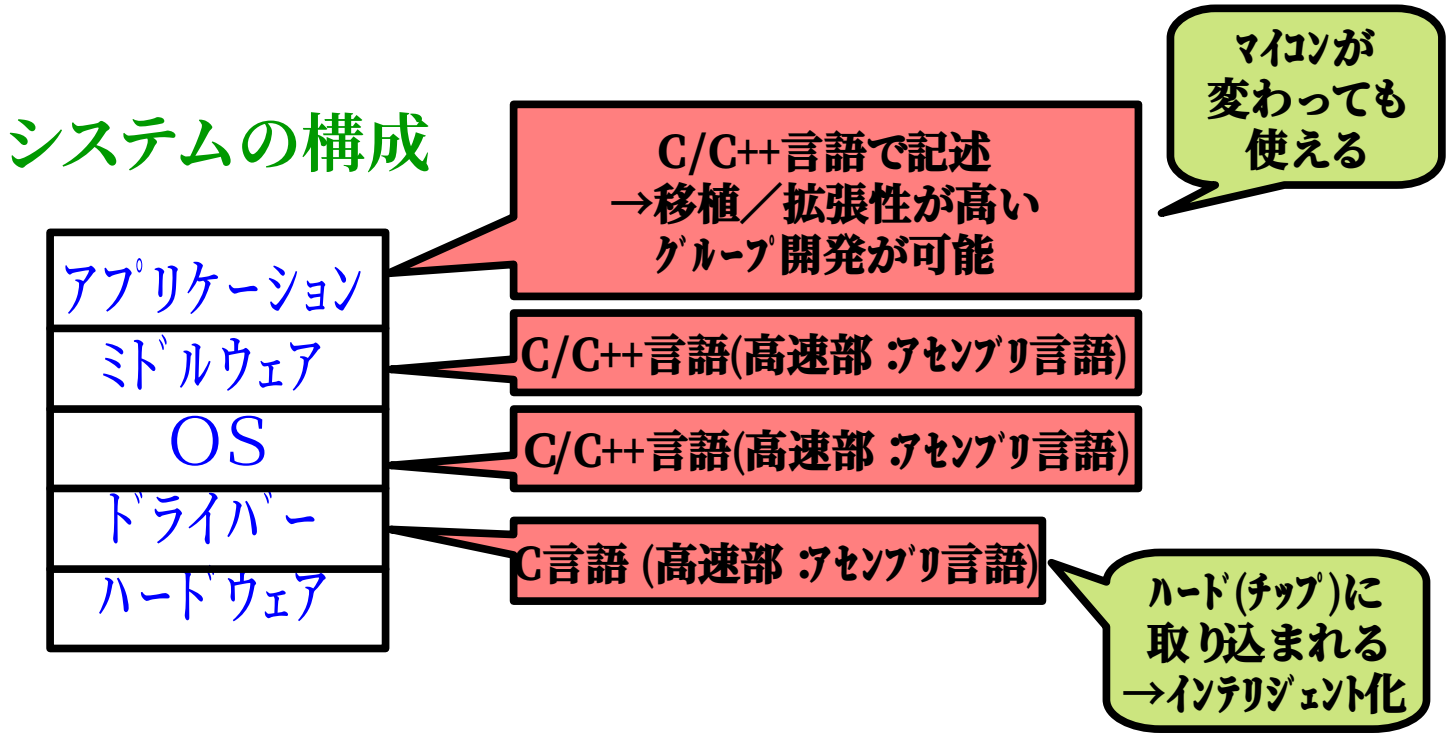
■保守時

- ●仕様変更や機能追加がタスク単位で対応できるため、システム全体に及ぼす影響が少ない。
- ●タスク単位でのコードが独立しているので、内部構造がすっきりし、コードの可読性が高い。
- ●タスク単位で独立しているため、他のシステムでもタスク単位で再利用できる。

OSのデメリット

- パフォーマンスの低下
  - タスク切り替えによる速度的なオーバーヘッドが発生
- コスト増
  - OSのためのメモリを確保
    - OSのコード/データ
    - タスク毎に必要なスタック（コンテキストの保存）
  - OSの導入コスト
- OSの選択
  - 目的に合致したOS 組込みLinux、ITRON、Windows CE...
  - OSを使いこなすための学習

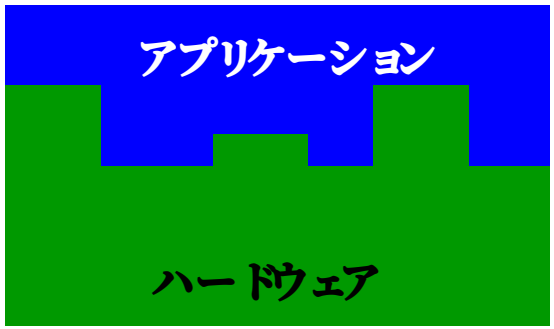
なぜOSを使うのか

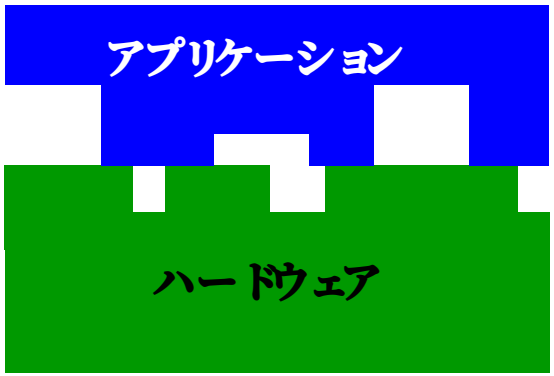


- ミドルウェアより上位を繰り返し使うためOSを使用
- ミドルウェアの流通化/共通化

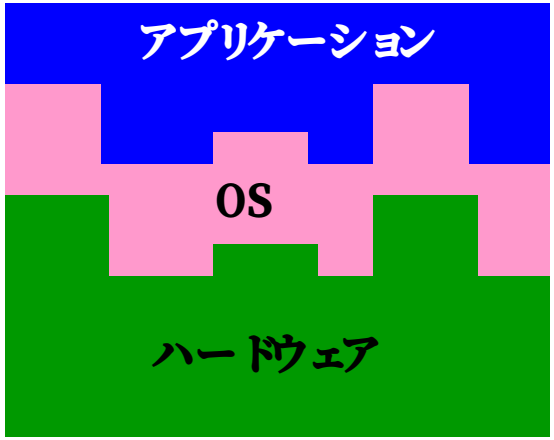
なぜOSを使うのか

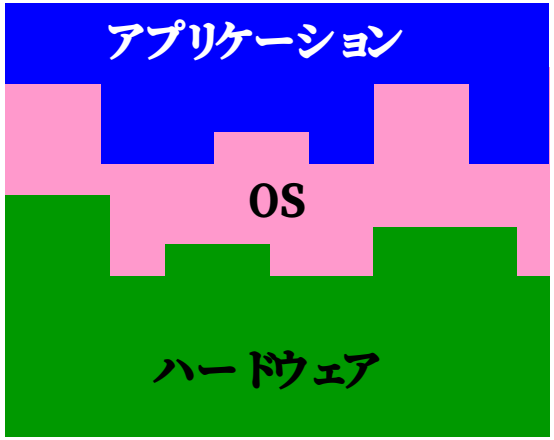
■OSを使わない場合





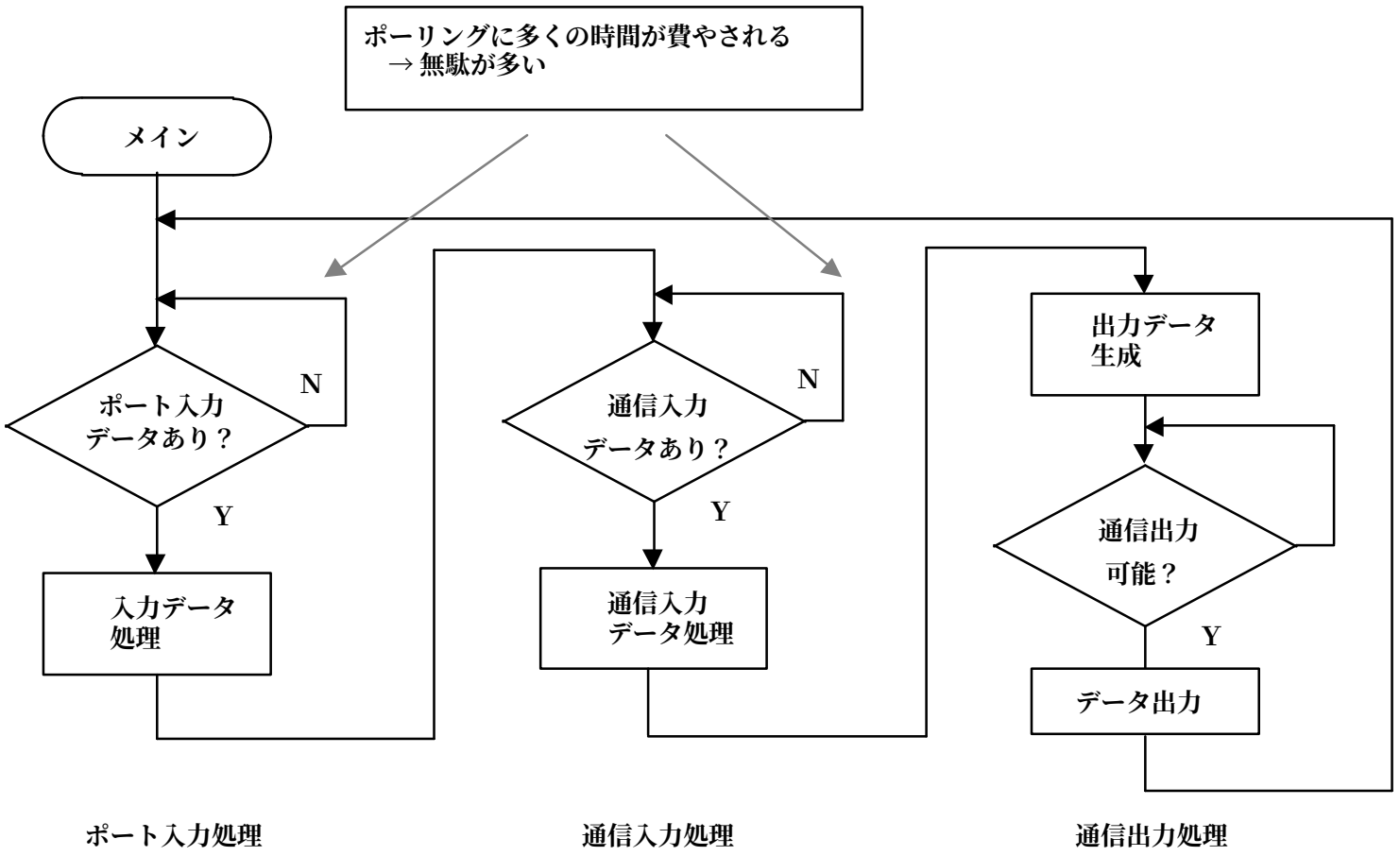
■OSを使う場合





OSの利用

■ OSなしのプログラム：一筆書き

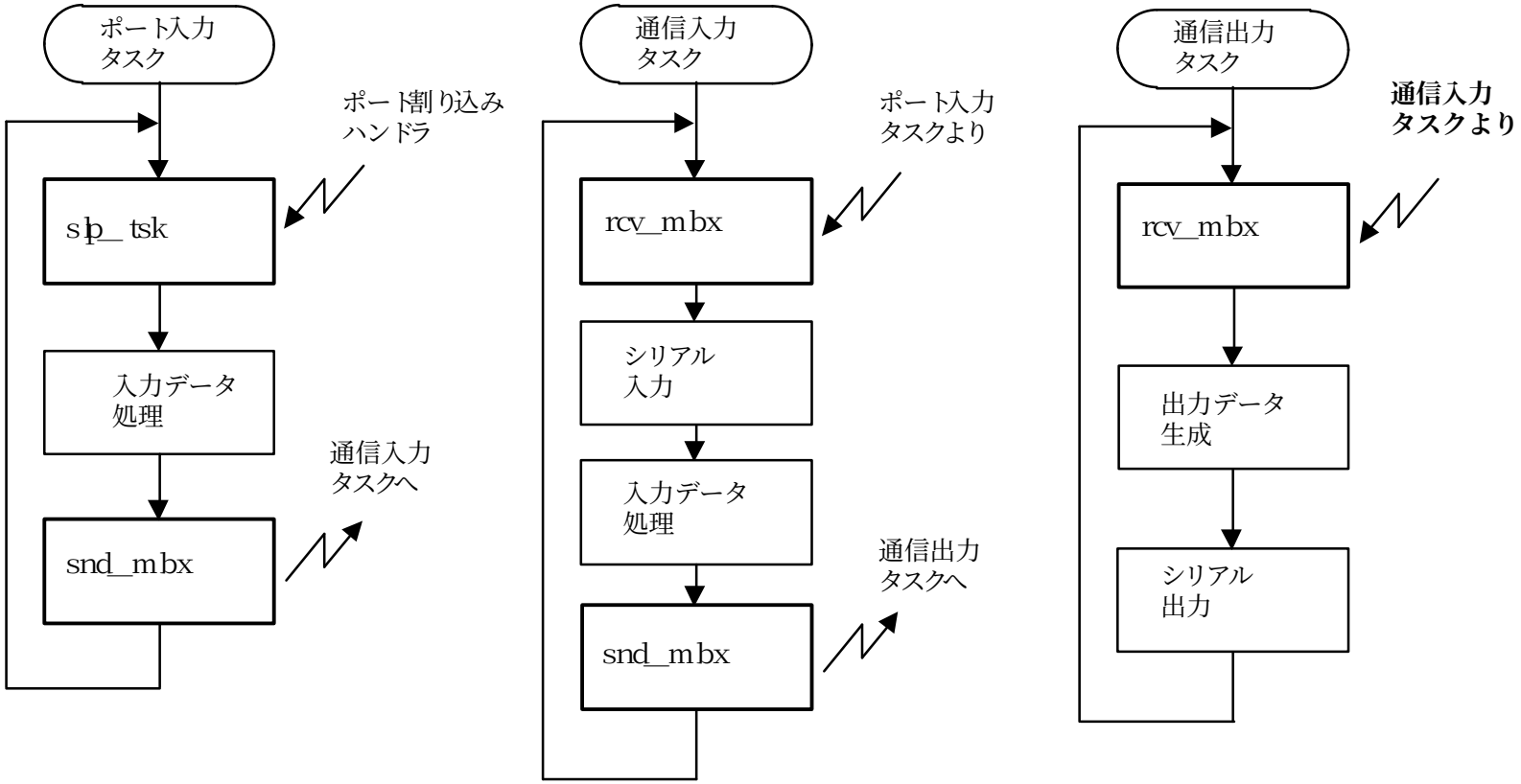


OSの利用

■ OSを使ったプログラム :仕様書に沿ったプログラミング、モジュール化

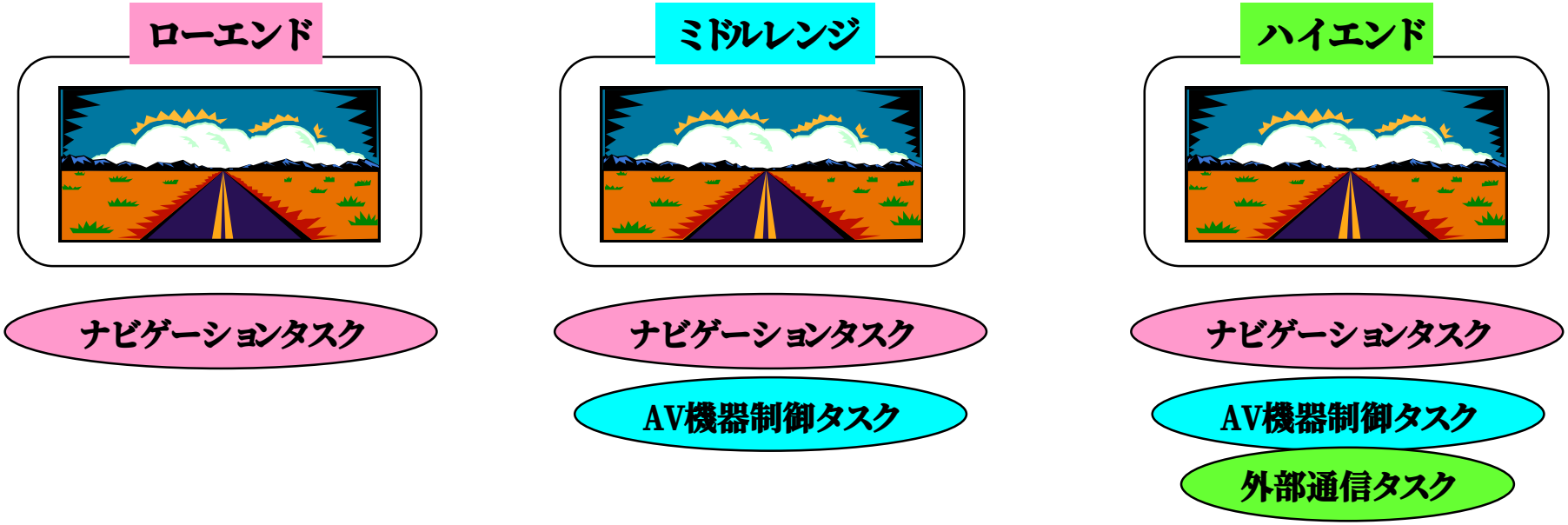
他のタスクとの同期をとる各種サービス
コールが用意されている

待ち時間はOS内で優先順位に従い
他のタスクに振り分けられる



OSの利用

- 仕様変更や機能追加がタスク単位で対応できるため、システム全体に及ぼす影響が少ない。
- タスク単位でのコードが独立しているので、内部構造がすっきりし、コードの可読性が高い。
- タスク単位で独立しているため、他のシステムでもタスク単位で再利用できる。



OSのコスト

- **オープンソース、ライセンスフリーのOSが注目されている**
  →Linux、ITRON
- **ただし、ライセンスフリーだからと言って、お金が一切かからないという訳でない。**
  - 問題は、誰が保証するのか？
  - だれが、バグを退治するのか？
  - だれが、メンテナンスするのか？
  - やはり、しっかりとメンテナンスができる会社と契約する必要がある。
- **上記のメンテナンス費用を考えると、台数によっては、他のOSの選択肢もある。**

代表的な組込みOS :ITRON

■ トロン協会で仕様を制定 :www.assoc.tron.org/jpn/tronproject/tp_p-11.html
■弱い標準化（実装は各社まかせ）
■仕様が公開されている
■自分で仕様に沿って作成するのであれば、ライセンスは不要。
■日本の組込みシステムでは、40-50%以上がITRONを使っている
■CPUメーカ製ITRON、3rdパーティ製あり
- ルネサス、NEC 、東芝、富士通等
- ミスポ社（NORTi）、イーソル社（PrKERNEL）等

■特徴
- フットプリントが小さい
- リアルタイム性がある

μITRON4.0仕様 :www.ertl.jp/ITRON/SPEC/mitron4-j.html　　μITRON4.0仕様書

代表的な組込みOS :TOPPERS/JSPカーネル

■ITRONからの分家

・TOPPERSプロジェクト(www.toppers.jp/)

正会員 :団体 (97)個人 (13)、準会員 (71)、特別会員 :団体 (9)個人 (8)が加盟(2006年3月現在)

■特徴

・開発環境まで含めてフリーソフトウェアのみで構築可能

・読みやすく改造しやすいコード

・他のターゲットへのポーティングが容易な構造

大部分をCで記入、ターゲット独立部とターゲット依存部を明確に分離

・高い実行性能と小さいRAM使用量

■LinuxホストやWindowsホスト上で動作シミレーションが可能

■レポートウェア :ライセンスはフリー。ただし、プロジェクトで使用の報告義務がある。

■T-Kernelとの相違

・TOPPERS :小さなシステム向き

・T-Kernel :より大規模なシステム向き

■開発成果物 :TOPPERS／JSPカーネル、TOPPERS／FI4カーネル、TOPPERS／OSECカーネル、C++APIテンプレートライブラリ、カーネルテストスイート、TNET (ICP／IPスタック)、モデルベース開発、リモートリンクローダ、ダイナミックローディングマネージャー、マルチプロセッサ対応リアルタイムOS、宇宙機向け高信頼リアルタイムOS、FlexRay対応TOPPERS／OSEC

代表的な組込みOS :T-Kernel

■ITRONからの発展系

T-Engineフォーラム(www.t-engine.org/japanese.html)
502団体団体(2006年3月現在)

■特徴

- T-Engine :ミドルウェア流通のプラットフォームを規定
- シングルワンソース
- スタティック／ダイナミックメモリアロケーションの 2つを持つ

■T-License

その条件で、無償で製品に組み込み可能。
レポートウェア :ライセンスはフリー。
ただし、プロジェクトで使用した場合の報告義務がある。

代表的な組込みOS 組込Linux 1

- オープンソース
- ロイヤリティフリー
- OSが標準でもつ機能が豊富
- ソフトウェア部品が数多くある
- マルチタスク、マルチユーザ
- モジュールをダイナミックに追加、削除、変更可能
- Linuxベンダー
  - MontaVista :MontaVista Linux
  - 日新システムズ :TimeSys Linux
  - アックス :axLinux
  - リネオソリューションズ :Lineo
  - その他、多数

組込みLinuxの仕様策定組織 :Emblix (www.emblix.org/)

代表的な組込みOS 組込Linux 2

■MMU機能があるCPUの場合

- ●サポートLinuxベンダー
  SH-Linux、MontaVistaLinux、axLinux、TimeSysLinux、Lineoなど
- ●サポートCPU :ARM、SH4など
- ●Linux PC上でアプリケーションの開発が可能。

■MMU機能がないCPUの場合

- ●サポートLinuxベンダー：
  axLinux（アックス）、uCLinux（uedhat）など
- ●サポートCPU :FRV、ARMなど
- ●レギュラーLinuxよりフットプリント小、高速起動

代表的な組込みOS :CE Linux

■デジタル家電機器向Linux

■CELinuxフォーラム(www.celinuxforum.org/)

- ●松下電器産業とソニー、日立製作所など国内の電気メーカーを中心とする8社によって2003年7月に設立。

■Linuxカーネル2.4.20をベースに開発

- ●起動時間 ･終了までの時間の短縮
- ●低消費電力化
- ●AV グラフィックスに対する機能検討
- ●応答時間などの リアルタイム性の向上
- ●必要とされるメモリー容量の削減
- ●セキュリティ強化

GPL

■GPL :General Public License

- GNUプロジェクトが提唱するフリーソフトウェアのライセンス形態。
- ソースも公開の義務あり。
- 使用、複製、変更、頒布が自由。
- ただし変更、改良されたソフトウェアもGPLに従う必要あり。
- 基本的に無保証。

■GPLの抵触

- ユーザアプリケーションをGNUライブラリを使用して開発すると、GNUライセンスにより、ソース公開の義務が発生。組込みシステムでは問題となる場合がある。
- NewlibというGPLに抵触しないものもある。
- カーネルがGNUのため、それと静的にリンクされる静的ドライバも公開の義務が発生する。
- このためデバッグ中は、静的ドライバで作成し、最終版では動的ドライバに直す方法が多く使われている。

※ 動的ドライバもGPLに抵触すると場合もあるため要注意

代表的な組込みOS :WindowsCE

- 豊富なミドルウェア、アプリケーション、プロトコルスタック
- OSが標準でもつ機能が豊富
- ソフトウェア部品が数多くある

  IE、Windows Media Player、USB、Bluetooth、
  日本語IMEなど

  web上から3rdパーティ製のドライバがダウンロード可能

Windows CEホームページ
(www.microsoft.com/japan/windows/embedded/ce50/default.mspx)

代表的な組込みOS :Symbian OS

- スマートフォン用オペレーティングシステム
- 1998年ｴﾘｸｿﾝ，ﾉｷｱ，ﾓﾄﾛｰﾗ，ｻｲｵﾝ社共同で設立
- 携帯電話向けに特化したOS
- 開発の短縮化が可能
- アセンブラ層によるハードウェア、UI部との結合が容易
- ワールワイドでのシャアが高くアプリケーションが豊富
- Nokia等の世界有数の携帯電話機メーカーがライセンス中。
  特に海外でのシェアが高い。
- 日本の携帯メーカもライセンス中。富士通／三菱等
- Ver6.1、Ver7.0x、Ver8.1b、Ver9.ｘ あり
- PC上でのシュミレーション環境が充実

symbian
Symbian OS - the mobile operating system

Symbianホームページ(www.symbian.com/Japan/)

代表的な組込みOS :OSE

■ENEA社（本社スウェーデン）の開発したOS、NOKIA、エリクソンの依頼により、携帯用に開発

エニアホームページ(www.enea.co.jp/)

■特徴

- フットプリント= μITRON並み、8～32ビットCPUをサポート
- 分散処理が可能、多くの通信プロトコルを持つ
- MMUをサポート、ハードリアルタイムを確保
- プロセスモデル、ダイナミックパッシングメッセージ
- イルミネータ:PC上で動作するデバッガ、ツール各種あり

代表的な組込みOS :メモリ容量比較

	ROMサイズ	起動時間
ITRON	400Kbyte	瞬時
MMU未使用Linux #1	0.8～1M bytes	5-6秒
MMU未使用Linux #2	1.4M bytes	
MMU使用 Linux	数M bytes以上	数十秒

代表的な組込みOS :比較表

	リアルタイム性	ソフトウェア部品の流通性	ソース公開	ライセンス	開発環境（コンパイラ）
ITRON	大	小	ベンダ依存	有料 or ライセンスフリー	Windowsベース（メーカ/3rdP製コンパイラ）
TOPPERS	大	コンポーネントWGあり	オープンソース	TOPPERSライセンスにて、無償配布可能	GNU環境（gccコンパイラ）
T-Kernel	大	中->大	オープンソース	T-Licenseにて無償配布可能	GNU環境（gccコンパイラ）
Linux	小 注：kernel2.6	大	オープンソース	GPLにて無償配布可能	GNU環境（gccコンパイラ）
Windows CE	小	大	ソース公開有り	有料	Platform Builder

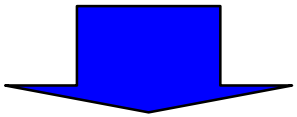
組み込みRTOSと情報系OS

■組み込みRTOS　TRON系、PrKernel

- μsec単位での割り込み応答が可能
- 組み込みハードウェアに特化
- ミドルウェアの不足
- ローパワーに向いている 携帯等）

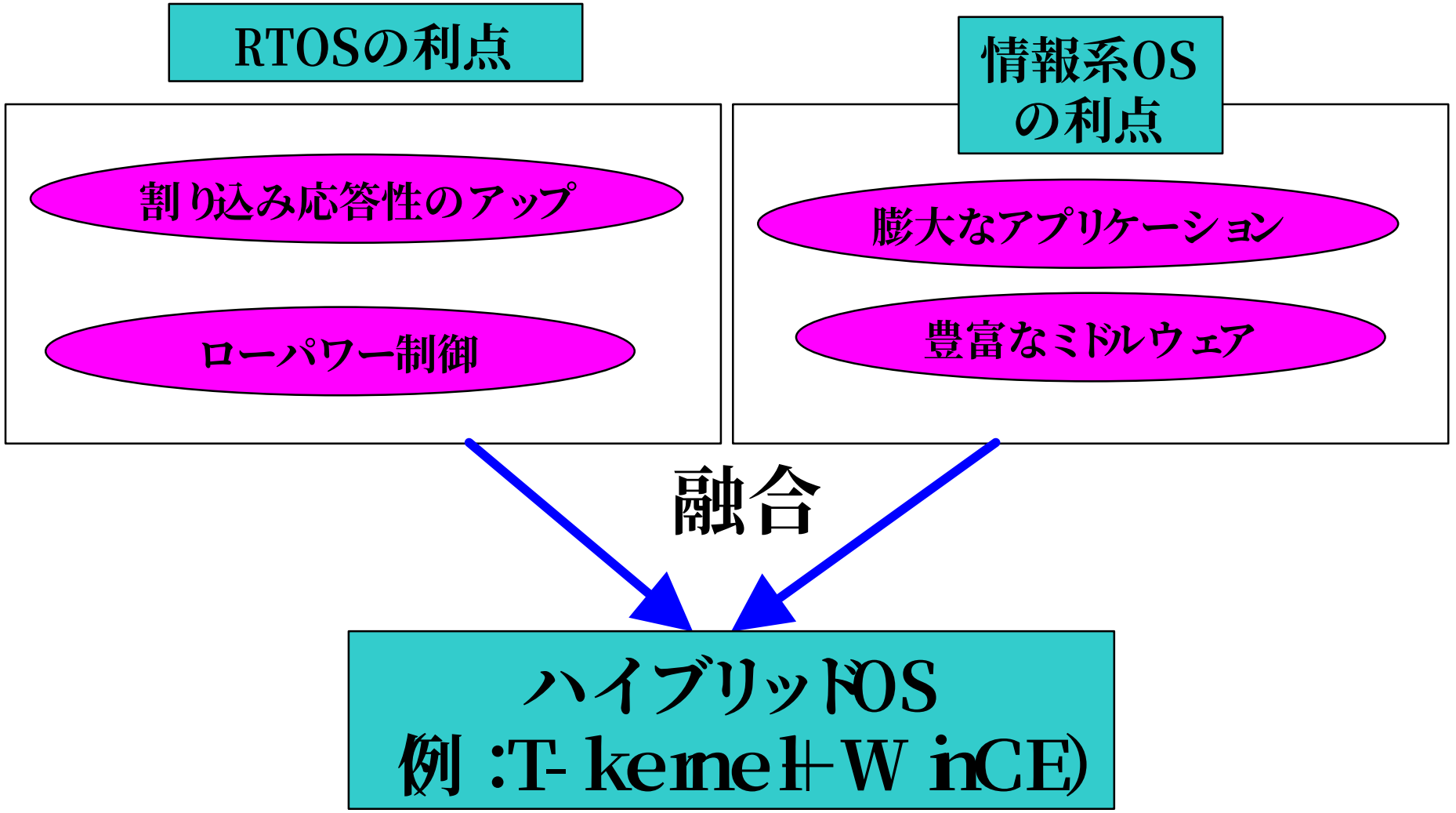
■情報系OS： Windows、WindowsCE、Linux

- 100msec以上の割り込み応答がかかる（Linuxの例）
- 膨大なアプリケーション
- 豊富なミドルウェア



良し悪しの問題でなく、用途、技術フォーカスが相違

それぞれの利点を生かす



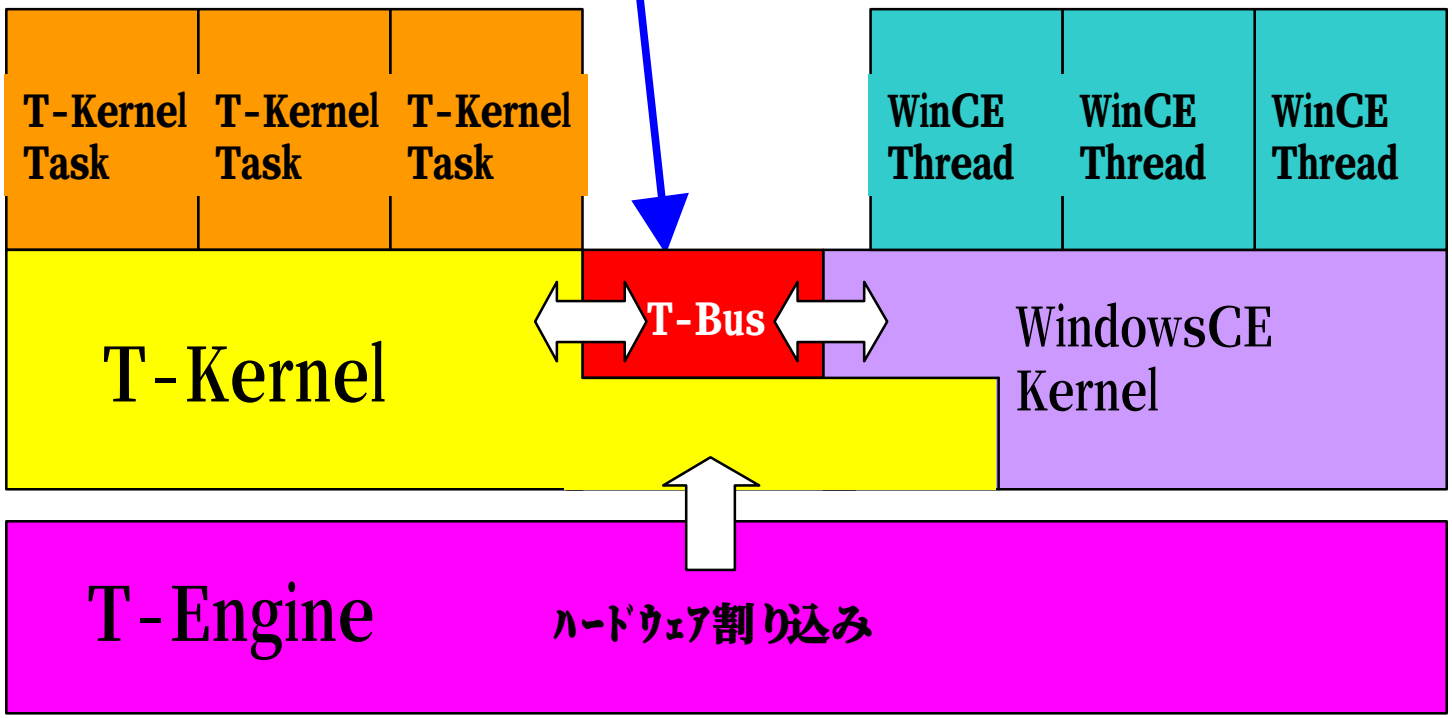
T-Engineと Windows CE

割り込み

スケジューリング

同期通信

- 割込みやスケジューリングは、T-Kernel側が優先的に行われる。
- 資源管理は基本的に、両者で分割する。



出典：2003/9/25 T-Engineフォーラム
記者発表PDF

OSの使い方

■それぞれのOSの得意分野があるので近年、各種OSを組み合わせ始めたハイブリッド化（不得意分野を補う目的）

- ITRON　組み込み用途に多い
- Linux　オープンソース、豊富なソフトウェア群
- Intent　JAVAが高速で動作
- その他　GUIが強いOS、通信に強いOS等

OSの組み合わせ

ハイブリッドOSシステム

■ 組込みOS概要
- ○ OSの役割
- ○ OSのメリット/デメリット
- ○ 組込みOSの種類および用途

■ **リアルタイムOS概要**
- ○ **リアルタイムOSとは**
- ○ **リアルタイムOSのメモリ管理、スケジューリング方法**

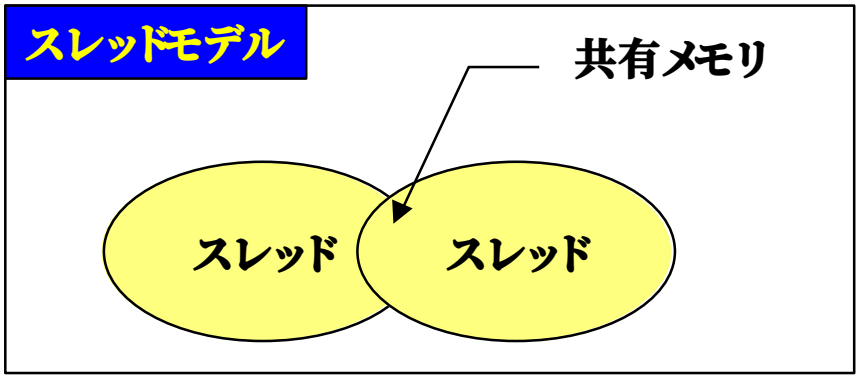
■ μITRON概要
- ○ μITRONの基本構造
- ○ μITRONの状態遷移

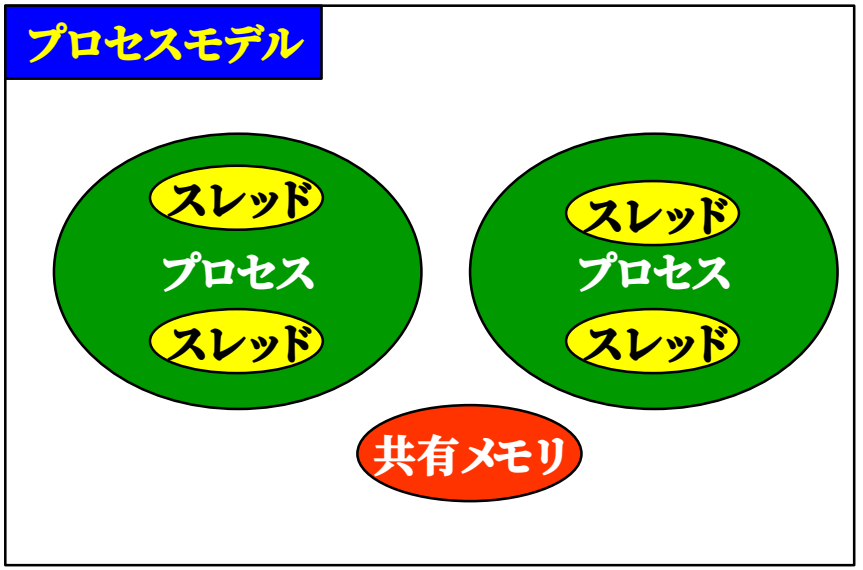
■ TOPPERS/JSPカーネル概要
- ○ TOPPERS/JSPカーネルのデータタイプ
- ○ TOPPERS/JSPカーネルの機能概要
- ○ TOPPERS/JSPカーネルによるプログラミング概要

OSの種類 :メモリ管理

■スレッドモデル
- •メモリ保護機能なし
- •MMUは使用しない
- •別スレッドのメモリにアクセス可能
- •ITRON, VxWorks, Nucleus, ……

■プロセスモデル
- •メモリ保護機能あり
- •MMUを使用
- •プロセス間で独立
- •プロセス内に複数のスレッド
- •Linux, UNIX, Windows, OS-9, ……

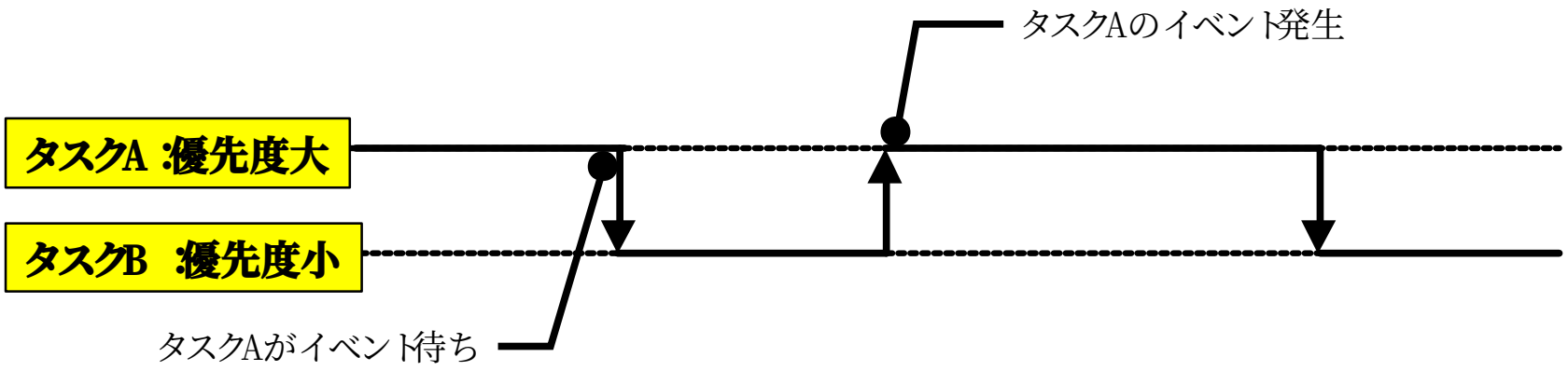




OSの種類 :スケジューリング方法

■イベントドリブン :事象駆動

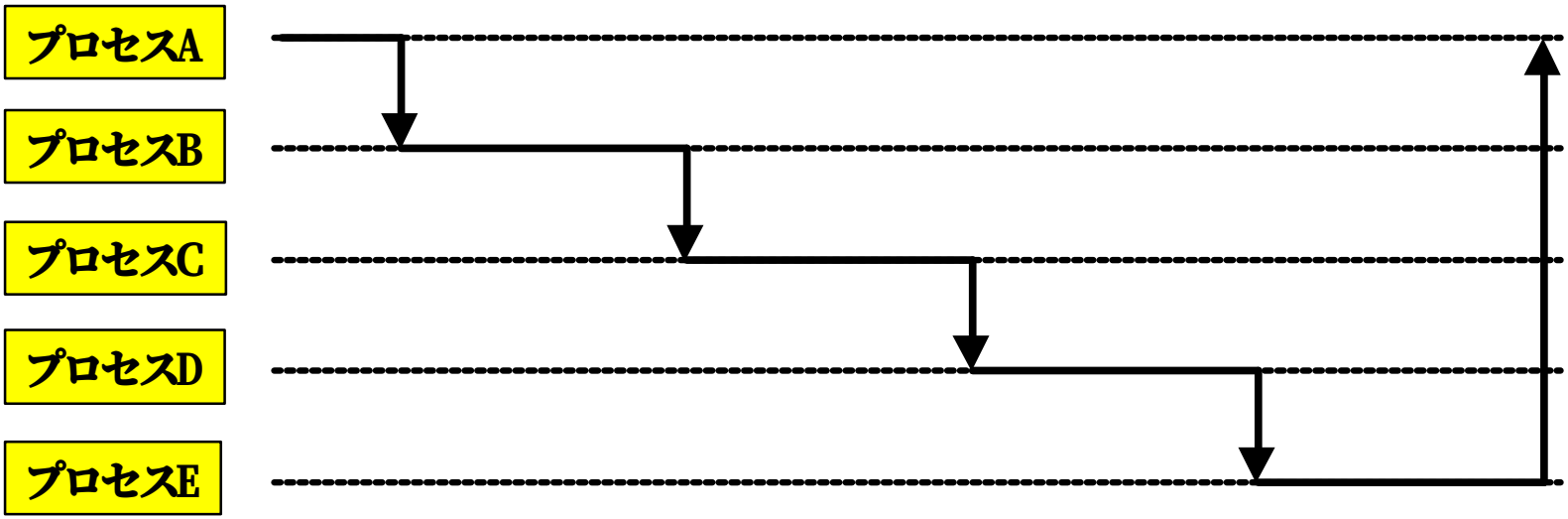
- イベントの発生によりタスクを切り替える方式
- タスクの優先度により実行 :プリエンプティブ



OSの種類 :スケジューリング方法

■TSS :タイムシェアリング

- 一定時間毎にタスクを切り替える方式
- 各タスクの実行は、時分割のためリアルタイム性はない
- プロセスはプリエンプトする機能を持っていない。



RTOSとは

■組込みシステム :リアルタイムシステム
- 決められた時間内に処理を終了させることが必要なシステム
- 応答時間に制約があるシステム
- 高速動作が必要な訳ではない

■RTOS
- リアルタイムシステムを構築するためのOS
- 基本的に、スレッドモデルでイベントドリブンのOSをRTOSと呼ぶ

■代表的なRTOS
- ITRON(TOPPERS, T-Kernel), OSE, VxWorks, Nucleusなど

■ハードリアルタイムとソフトリアルタイム
- ハードリアルタイム :時間制約を厳守しなければならない処理
  → カーナビ、ルータなど
- ソフトリアルタイム :時間制約が多少遅れても問題にならない処理
  → 自動販売機、エアコンなど

組込みLinux

■組込みLinuxとは

- マルチタスクや仮想メモリ、ファイルシステムなどの機能を含んだ汎用OSであるLinuxを、組込みシステム向けにカスタマイズしたOS

■組込みLinuxのメリット

- オープンソース
- ロイヤリティフリー
- 豊富なデバイスドライバ
- 豊富なミドルウェア
- ネットワーク機能が充実
- 安定動作
- アプリケーション開発に専念できる

組込みLinuxとPC用Linuxとの違い

■RTOSとは

●組込みシステム＝リアルタイムシステム

- 決められた時間内に処理を終了させることが必要なシステム
- 応答時間に制約があるシステム
- 高速動作が必要な訳ではない

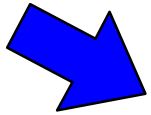
●RTOS

- リアルタイムシステムを構築するためのOS
- 基本的にスレッドモデルでイベントドリブンのOSをRTOSと呼ぶ

■PC用Linux

●TSS(タイムシェアリング)

- CPUの処理時間を一定の単位で分割
- 複数のアプリケーションを並列動作



PC用LinuxはRTOSではない

組込み用リアルタイムLinux

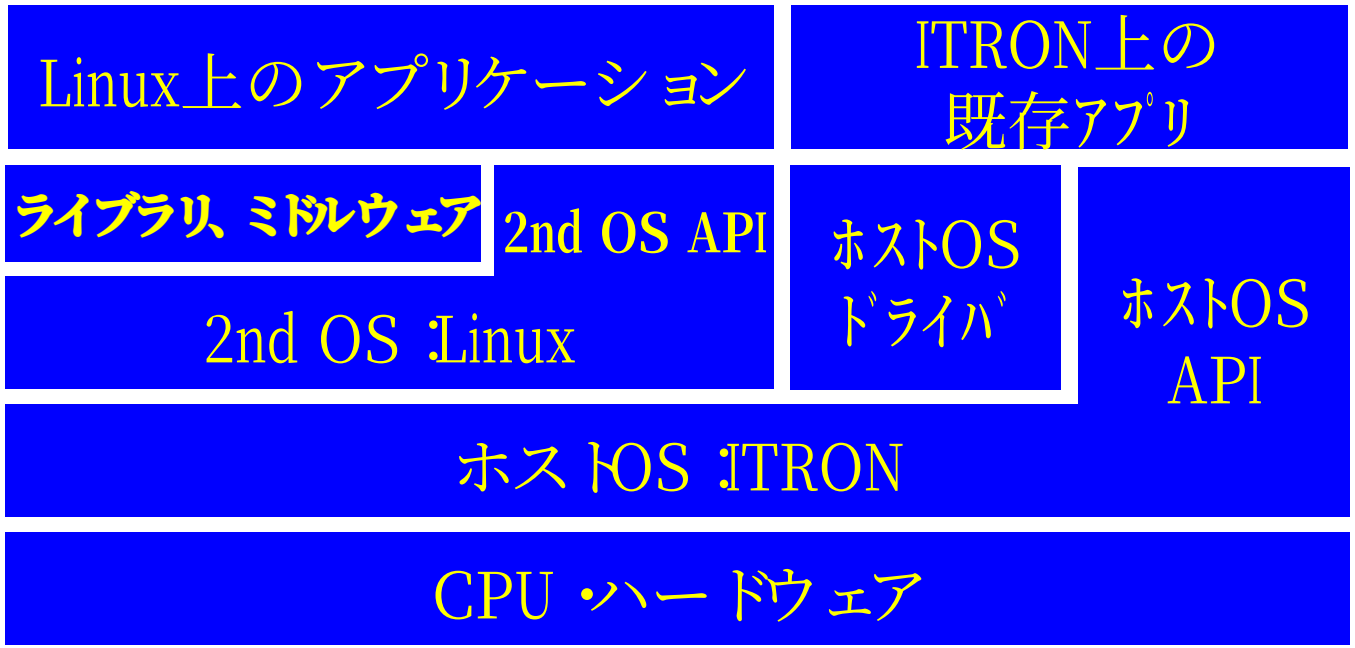
■ リアルタイムLinux

- Linuxカーネルにパッチを当ててリアルタイム処理を可能としたLinux
- 単純なTSSは行われない
- TSSに優先度をつけてスケジューリング
- Linuxカーネル自体を最適化してプリエンプティブ可能なカーネルに改造
- 種類
  - TimeSys Linux
  - SH-Linux
  - A&A Linux
  - ART-Linux

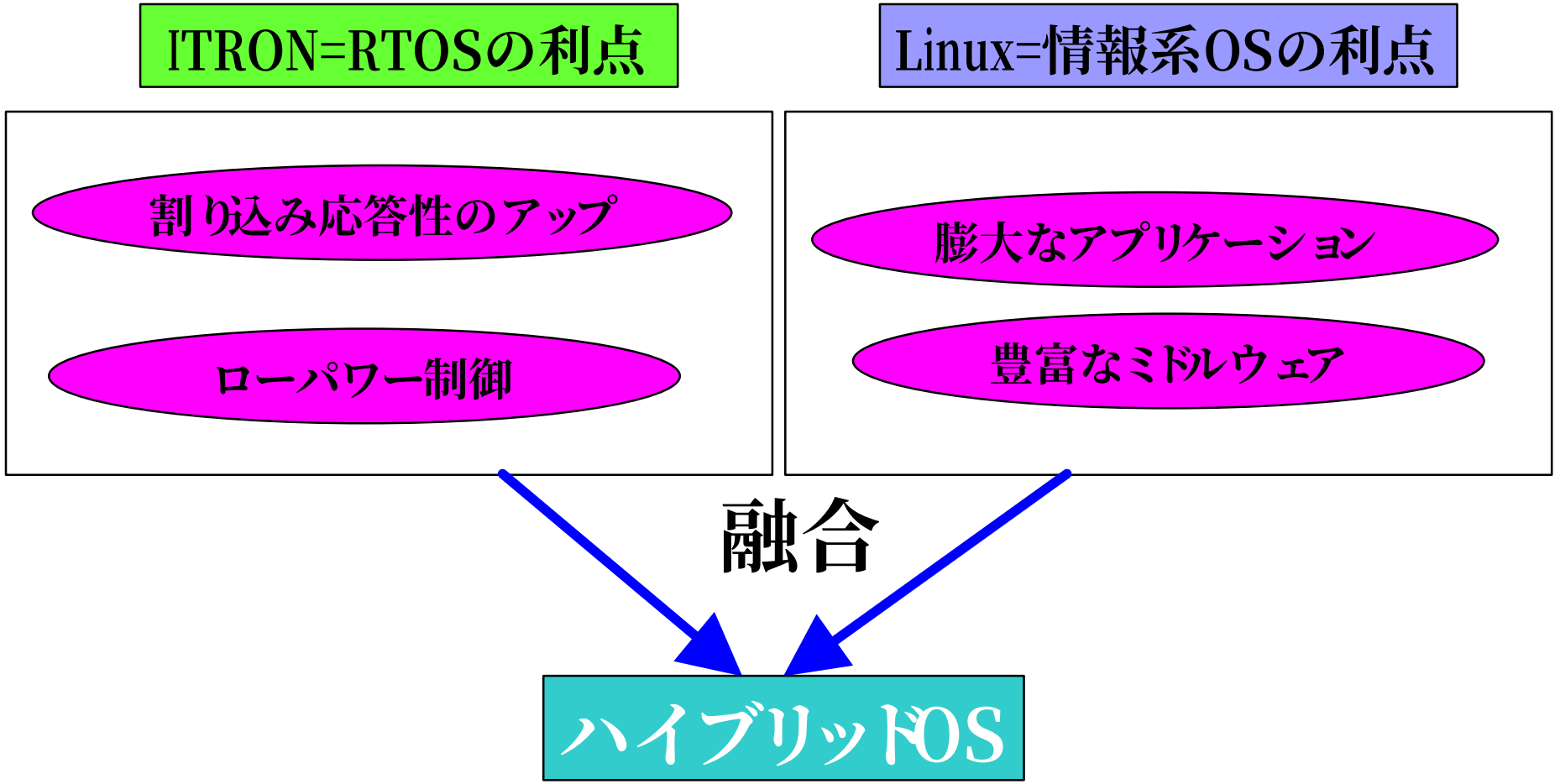
■ ハイブリッドOS

- LinuxとRTOSをハイブリッド化
- Linux on ITRON

Linux on ITRONの構造



Linux on ITRONの利点



組込みシステムでLinuxを使う利点・欠点

■利点

- ロイヤリティフリー
- 豊富なソフトウェア群：ミドルウェア、デバイスドライバ
- ネットワークに対応
- アプリケーションであれば、特に組込み機器としての意識は必要ない
- 高級言語を使えるエンジニアならば、すぐにアプリケーションが組める
- 安定動作
- 開発環境も同時に使える：コンパイラ、エディタ、デバッガ…

■欠点

- ワークステーションエンジニアと同じスキルが必要
- 一般の組込みエンジニアはワークステーションの知識がない
- Linux自体の構築
- デバイスドライバは誰が開発するか → CPUポーティング、ボードポーティング
- カーネルは誰が改良するか → 保証されない
- OSが要求するハードウェア資源が大きい
- リアルタイム性の問題
- ライセンス → ソースコードの公開

■ 組込みOS概要
- ○ OSの役割
- ○ OSのメリット/デメリット
- ○ 組込みOSの種類および用途

■ リアルタイムOS概要
- ○ リアルタイムOSとは
- ○ リアルタイムOSのメモリ管理、スケジューリング方法

■ **μITRON概要**
- **○ μITRONの基本構造**
- **○ μITRONの状態遷移**

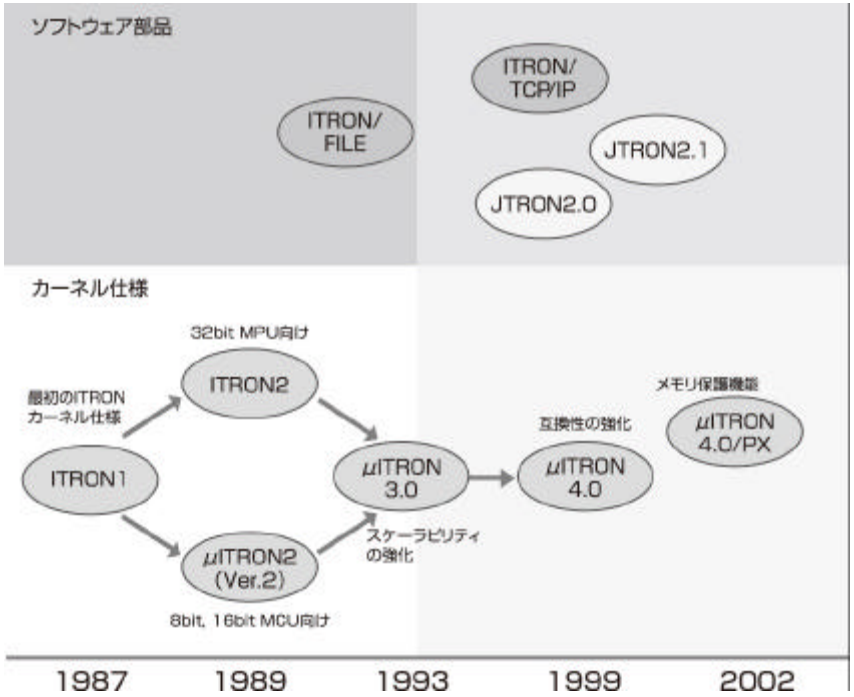
■ TOPPERS/JSPカーネル概要
- ○ TOPPERS/JSPカーネルのデータタイプ
- ○ TOPPERS/JSPカーネルの機能概要
- ○ TOPPERS/JSPカーネルによるプログラミング概要

µTRONとは

■TRON :The Real-time Operating system Nucleus
■ITRON :Industrial The Real-Time Operating system Nucleus
■ µTRON :Micro Industrial TRON

■概要

・1984年に東京大学坂村教授によって提案される
・TRON協会が仕様を決定
・AP仕様はオープンになっている
→ 各社の実装により動作に
多少が違いがある
・システムコールやAP名称などが
決められている
→ 基本的な部分のみ互換



リアルタイムOS

■ リアルタイムシステム

- 予め定められた時間的な制約があるシステム

例 :エンジン制御

エンジンが1回転する毎に処理し燃料を噴射する必要がある

→ **処理が間に合わないということは許されない**

■ リアルタイムOS

- リアルタイムシステムで使われるOS
- OSの提供するサービスにかかる時間が予めわかっているOS

■ リアルタイムカーネル

- リアルタイムOSのためのカーネル（中核モジュール）
- **μTRONはリアルタイムカーネルである**

リアルタイムカーネル

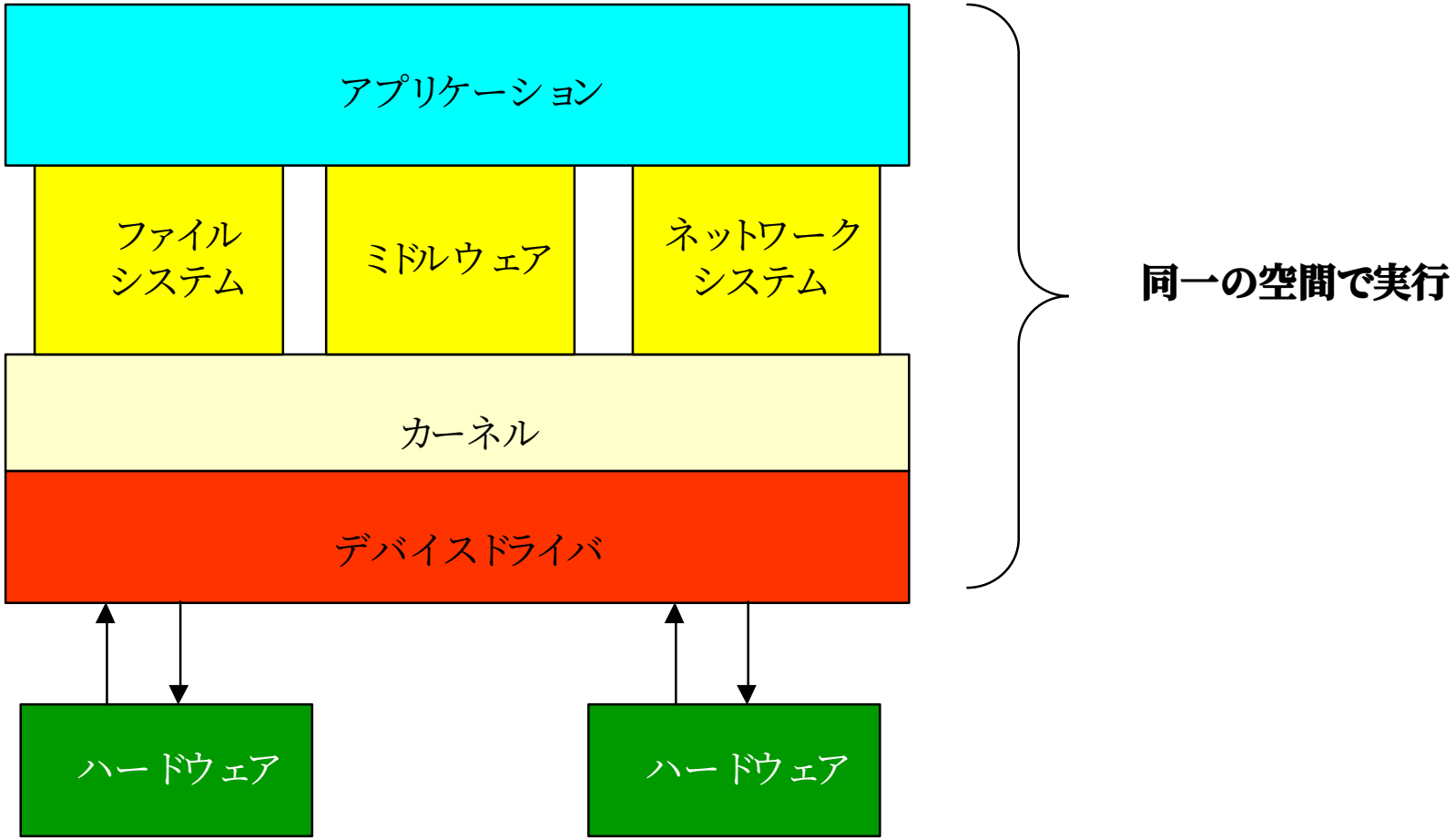
■ リアルタイムカーネルの機能

- マルチタスク :資源(CPU、内部I/O、メモリ等)の仮想化 •多重化
- タスク間通信と同期
- 割り込み管理/処理
- 例外管理/処理
- 時間同期/管理
- メモリ管理
- システム管理など

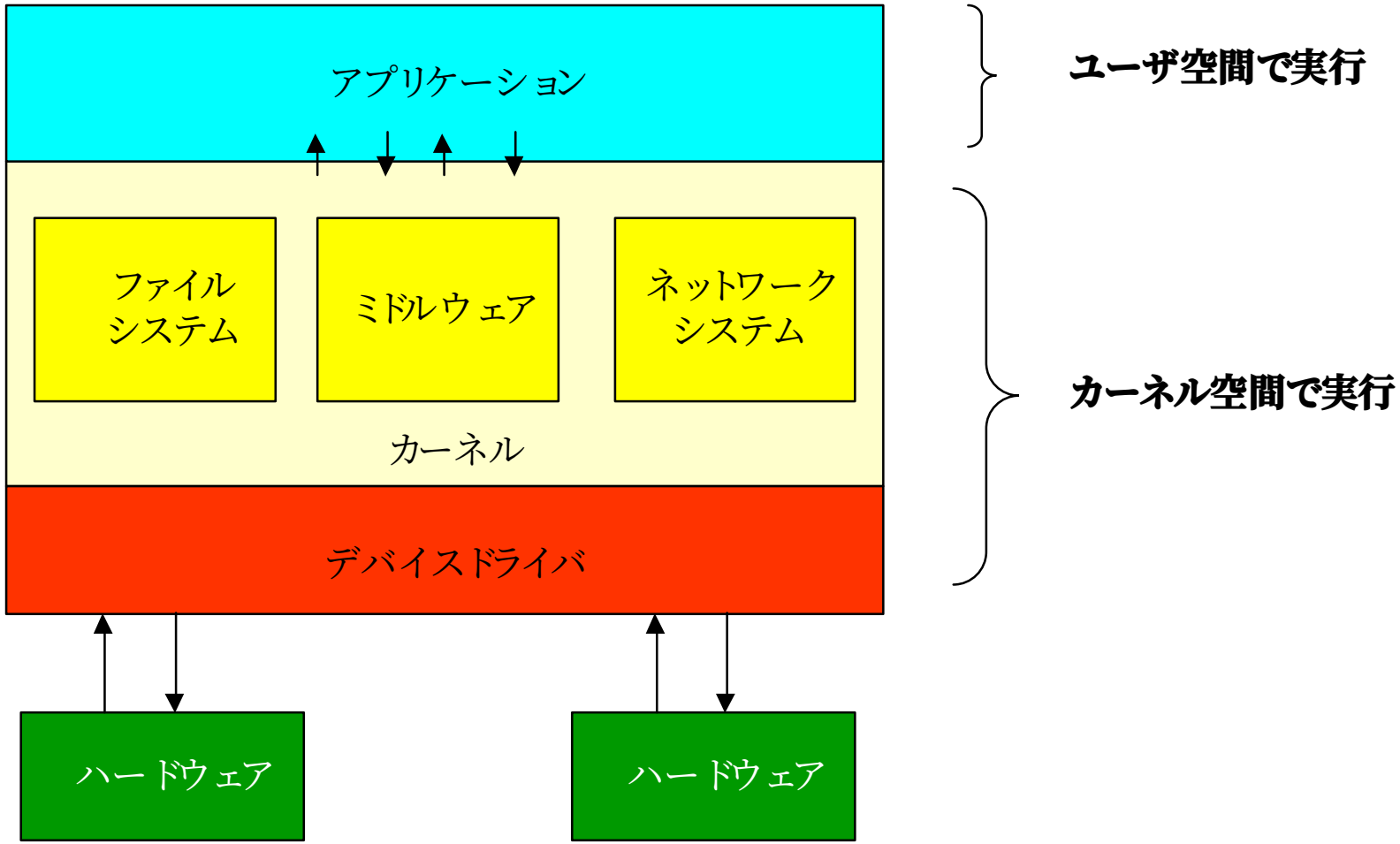
■ シェルの機能

- 入出管理
- 通信 •ネットワーク
- ユーザインターフェイスなど

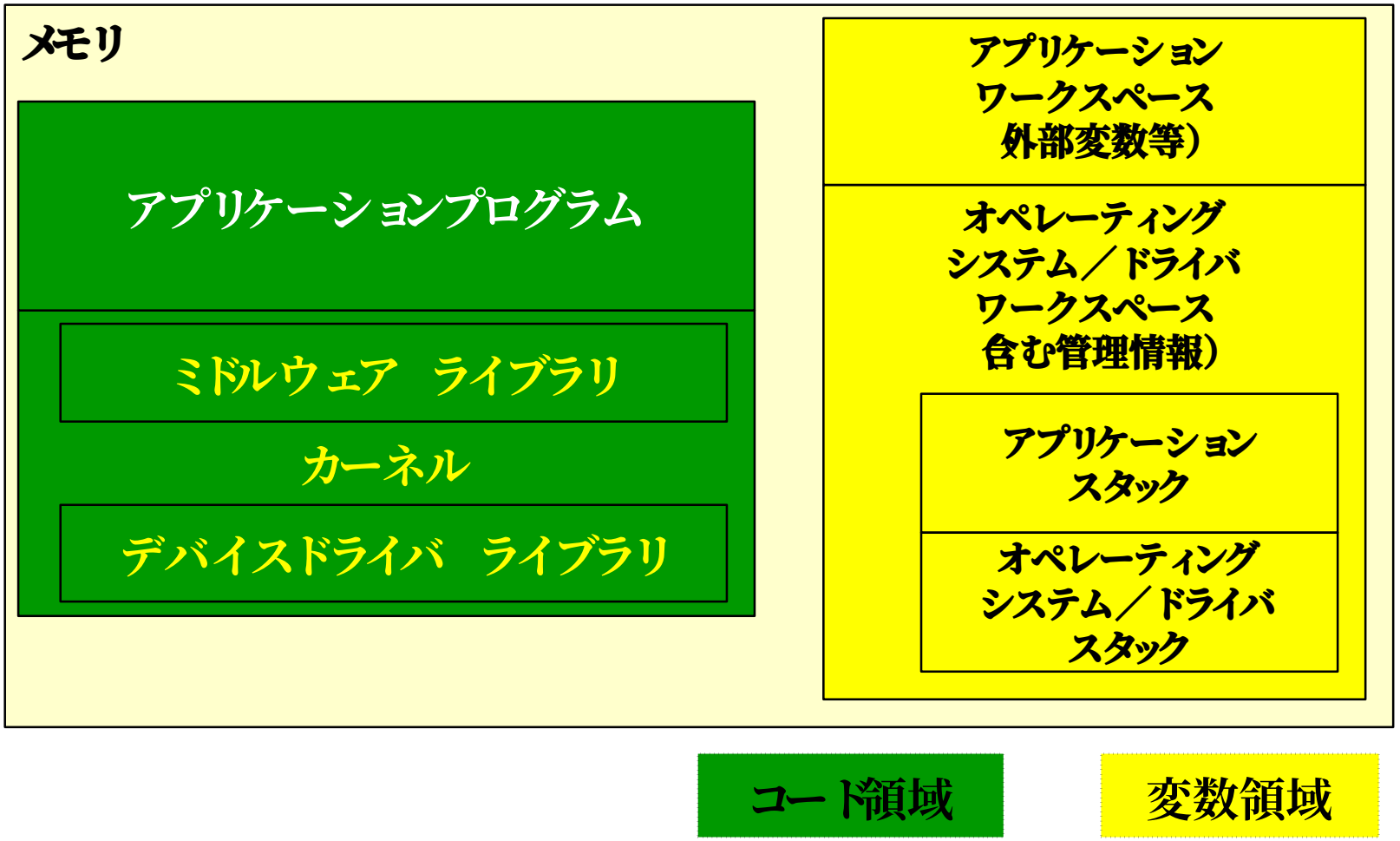
μTRONの基本構造



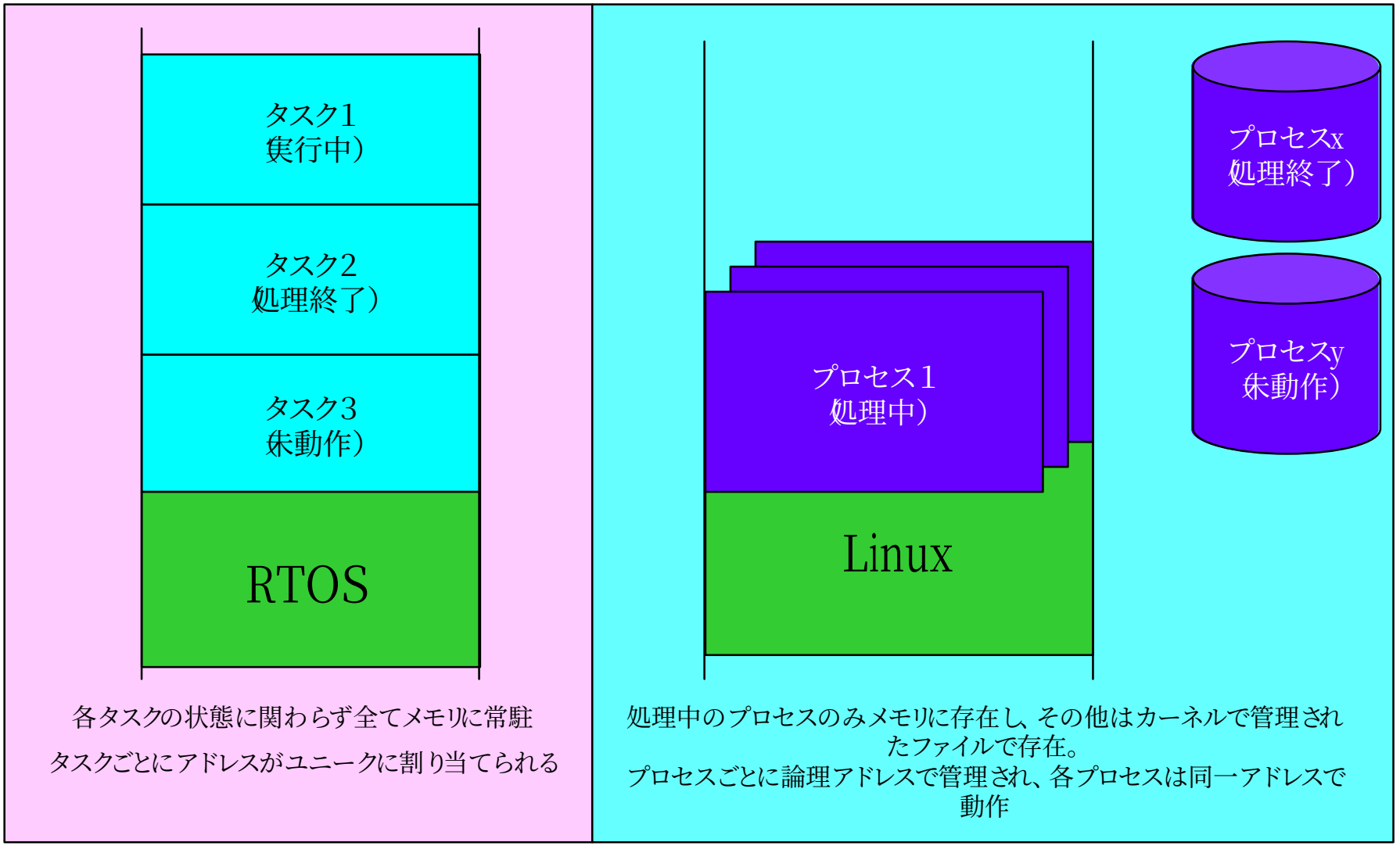
Linuxの基本構造



カーネルとデバイスドライバ、ミドルウェアの関係



メモリ管理



各タスクの状態に関わらず全てメモリに常駐
タスクごとにアドレスがユニークに割り当てられる

処理中のプロセスのみメモリに存在し、その他はカーネルで管理されたファイルで存在。
プロセスごとに論理アドレスで管理され、各プロセスは同一アドレスで動作

基本的な用語 :1

■タスク

- 並列で実行するプログラムの単位
- 実際に実行されるタスクは 1つのみ
- タスク切り替えによりマルチタスクを実現
  - → スレッド:メモリ保護機能がないOSの タスク
  - → プロセス :メモリ保護機能があるOSの タスク

■ディスパッチ

- タスクを切り替えて (タスクスイッチ) 実行状態へ遷移
- ディスパッチャ :タスクを切り替える処理を行う機構

■スケジューリング

- 次に実行するタスクを決めること
- スケジューラ :次に実行するタスクを決める機構

■コンテキスト

- ある時点でプログラム を実行する環境 (広義)
- どのスタック領域を使うか (狭義)
- レジスタ類の退避/復帰

■優先度 (プライオリティ)

- 処理を実行する順序を決める

基本的な用語 2

■サービスコール

•アプリケーションプログラムからカーネルまたソフトウェア部品を呼び出すインターフェース

■システムコール

•カーネルのサービスコール

■オブジェクト

•カーネルやソフトウェア部品が操作の対象とする資源

■ID番号

•オブジェクトを識別する番号(連番)

■優先度

•タスクやメッセージなどの処理順序を制御するために、アプリケーションによって与えられたパラメータ

○タスク優先度 :1～16の16段階

○メッセージ優先度 :タスク優先度以上の段階をサポート

■タスク付属同期機能

•タスクの状態を操作して起床待ち、起床など、タスク間で同期(待ち合わせ)をおこなう機能

•起床待ち、起床、自タスク遅延などがある

■同期・通信機能

•タスクとは独立したオブジェクトにより、タスク間の同期・通信を行う機能

•セマフォ、イベントフラグ、データキュー、メールボックスなどにより通信を行う

タスクの状態

■実行状態(RUNNING)

CPUが割り当てられタスクが実行されている状態

■実行可能状態(READY)

高優先度タスクまたは同一優先度で先に実行可能状態となったタスクが実行状態のため実行できない状態

■待ち状態(WAITING)

条件が成立するまでタスク自身が実行を中断した状態

■強制待ち状態(SUSPENDED)

他のタスクにより強制的に実行を中断された状態

■二重待ち状態(WAITING-SUSPENDED)

待ち状態と強制待ち状態が重なった状態

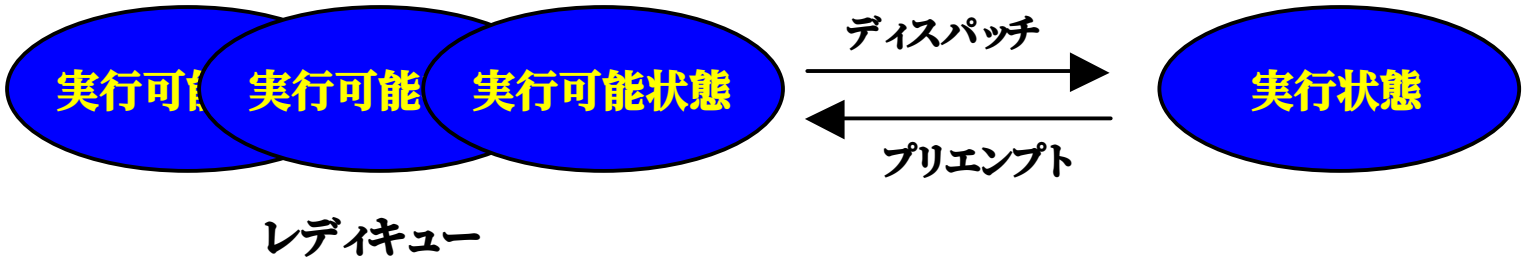
■休止状態 (DORMANT)

タスクが休止または終了した状態

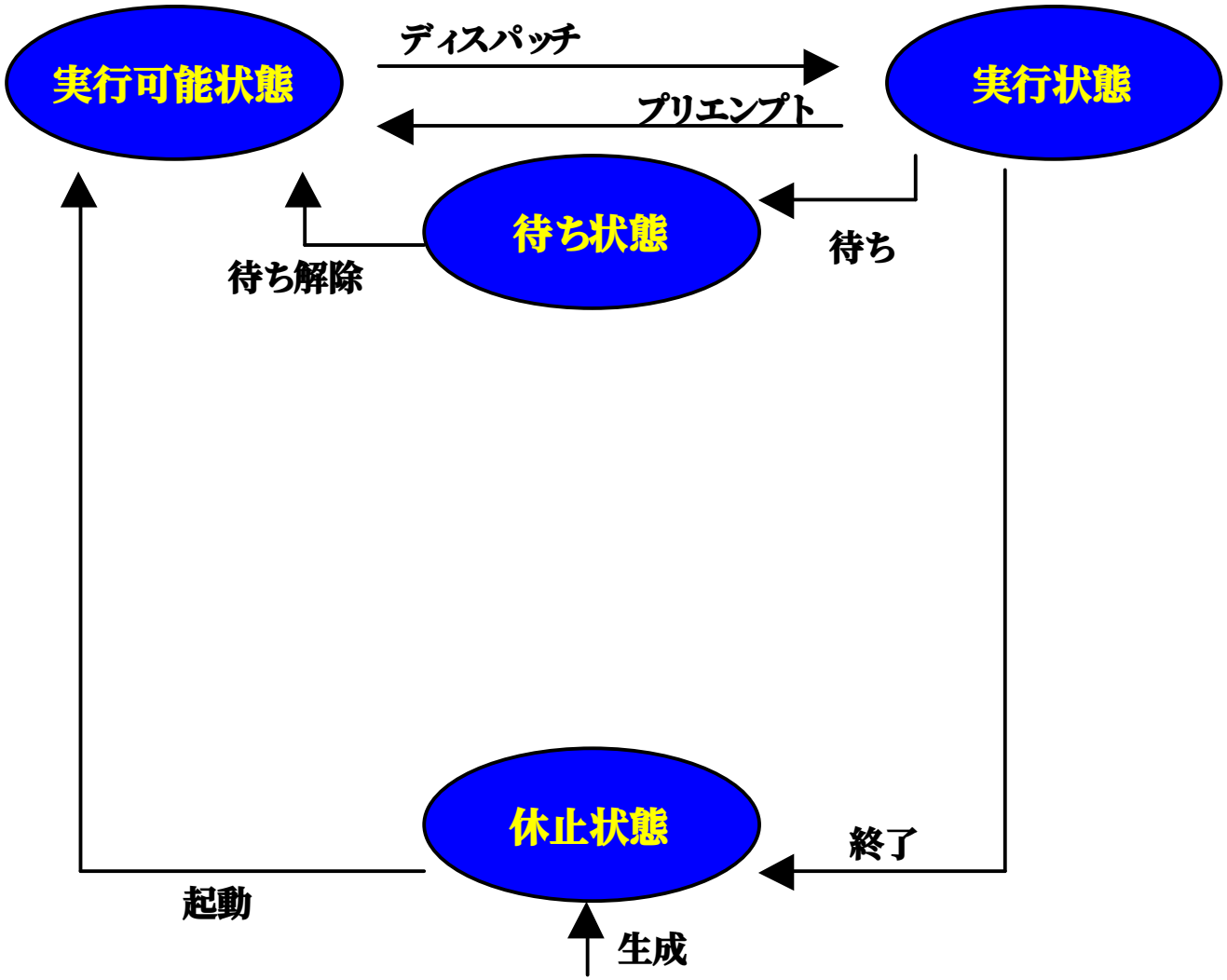
■未登録状態(NON-EXISTENT)

タスクが未生成または削除されシステムに存在しない状態

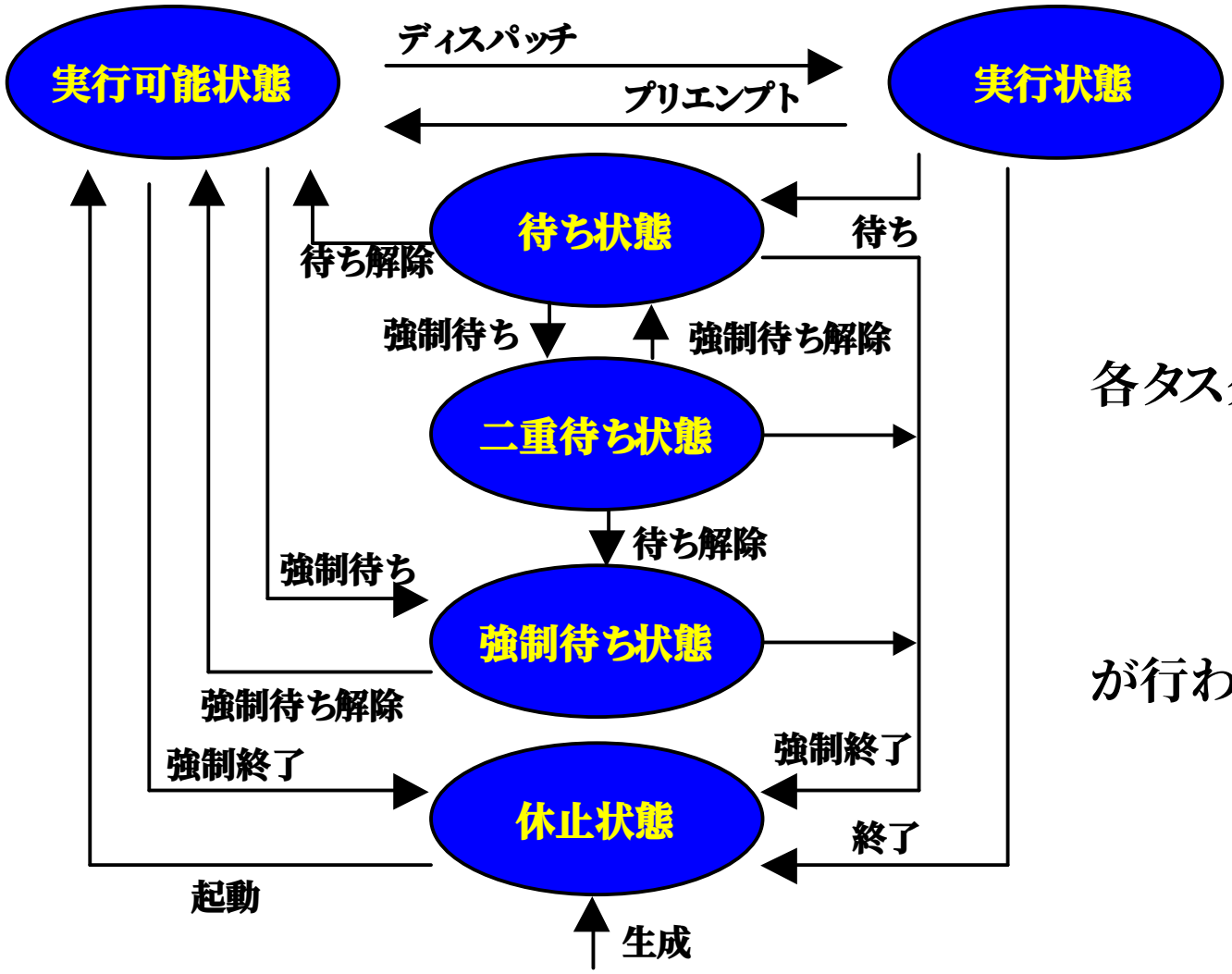
µTRONの状態遷移



μTRONの状態遷移

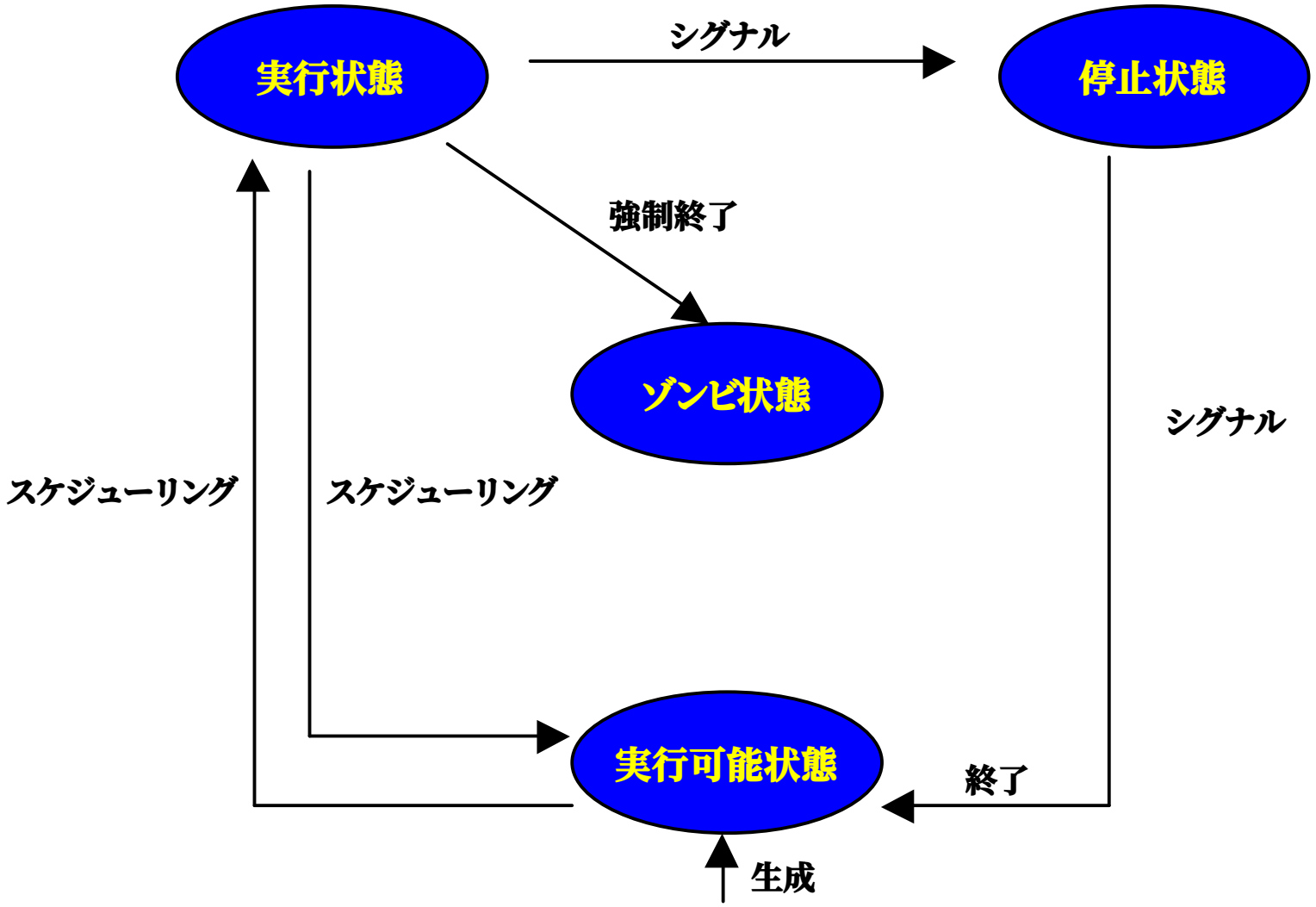


μTRONの状態遷移



各タスクはOS(μTRON)により
起動
中断
再開
終了
が行われる。

Linuxの状態遷移



ハードウェア資源へのアクセス

■ μTRON

- カーネルもアプリケーションも同一の空間で動作している。
- μTRONでは直接、ハードウェア資源を利用できる。
- アプリケーションから直接デバイスドライバを呼び出せる。

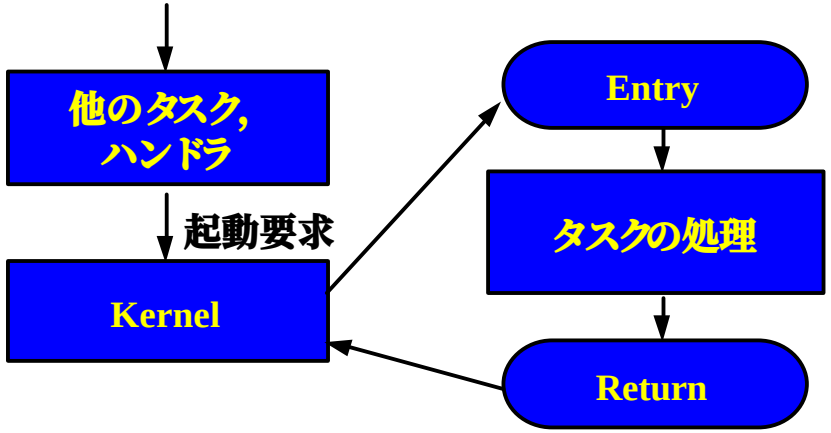
■ Linux

- カーネルとアプリケーションが別々の空間で動作している。
- Linuxでは直接、ハードウェア資源を利用できない。
- システムコールを使ってデバイスドライバを呼び出す。

基本的なタスク動作パターン

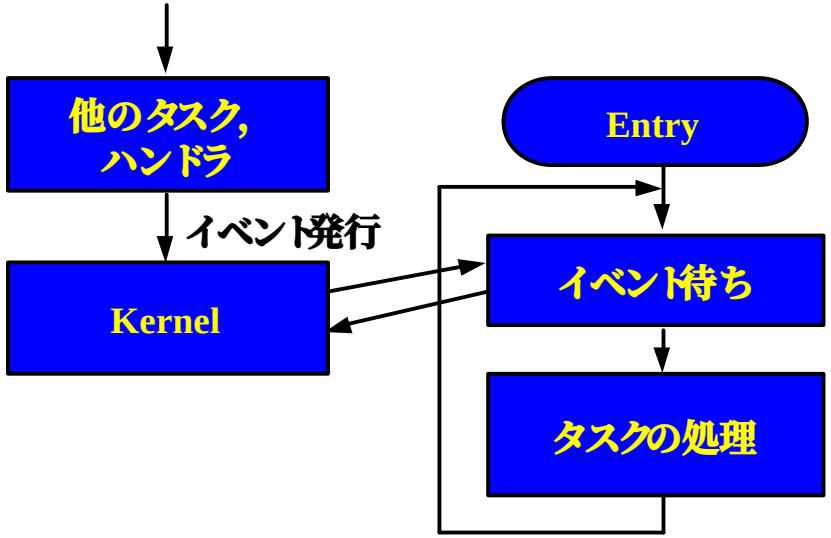
■タスク起動型

休止状態から他のタスクや
ハンドラにより起動され、
処理が終わると
休止状態に戻るパターン



■イベント駆動型

タスクは、1度実行され
何らかの待ち状態にあり、
他のタスクやハンドラから待ち
を解除されるイベントを受けて
動作し、処理が終わるとまた、
待ち状態に戻るパターン



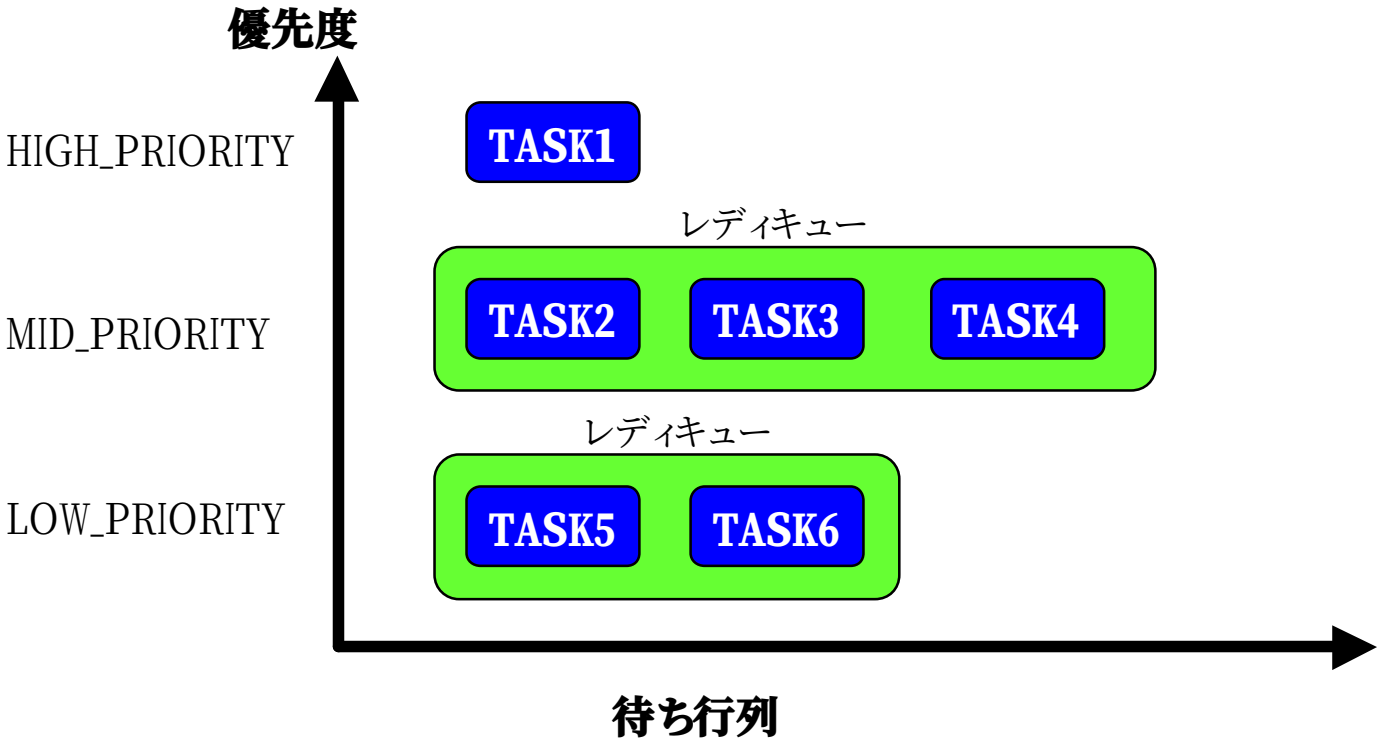
スケジューリング方式

■FCFS (First Come First Service)方式

最初に実行可能状態になったタスクから実行

■優先度方式

優先度の高いタスクから実行



- 組込みOS概要
  - ○ OSの役割
  - ○ OSのメリット/デメリット
  - ○ 組込みOSの種類および用途
- リアルタイムOS概要
  - ○ リアルタイムOSとは
  - ○ リアルタイムOSのメモリ管理、スケジューリング方法
- μITRON概要
  - ○ μITRONの基本構造
  - ○ μITRONの状態遷移
- **TOPPERS/JSPカーネル概要**
  - **○ TOPPERS/JSPカーネルのデータタイプ**
  - **○ TOPPERS/JSPカーネルの機能概要**
  - **○ TOPPERS/JSPカーネルによるプログラミング概要**

TOPPERS/JSPカーネルとは

■TOPPERS

TOPPERS（**T**oyohashi **OP**en **P**latform for **E**mbedded **R**eal-time **S**ystems）プロジェクトは、ITRON仕様の技術開発成果を出発点として、組込みシステム構築の基盤となる各種のソフトウェアを開発し、良質なオープンソースソフトウェアとして公開することで、組込みシステム技術と産業の振興を図ることを目的としたプロジェクトです。また、教育コースや教材の開発と、それを用いた教育の場を提供するなどの活動を通じて、組込みシステム技術者の育成に貢献することも目的としています。TOPPERSプロジェクトは、2003年9月に設立した特定非営利活動法人（NPO法人）を中心に、名古屋大学教授の高田広章をリーダとして、産学官の団体と個人の連携により推進しています。(http://www.toppers.jp/project.html より抜粋)

■TOPPERSカーネル

・TOPPERS/JSPカーネル(http://www.toppers.jp/jsp-kernel.html)

μITRON4.0仕様のスタンダードプロファイル（JSP：Just Standard Profile）規定に従って実装されたカーネル

・TOPPERS/FI4カーネル(http://www.toppers.jp/fi4-kernel.html)

μITRON4.0仕様のFI4(Fullset ITRON4)規定に従って実装されたカーネル

・TOPPERS/OSEKカーネル(http://www.toppers.jp/osek-os.html)

OSEK/VDX仕様Version2.2.1 ECC-2に準拠した自動車制御用リアルタイムOS

データタイプ

■signed int型に定義しているデータ型

・INT	符号付き整数
・BOOL	真偽値
・FN	機能コード
・ER	エラーコード
・ID	ID番号
・PRI	優先度
・TMO	タイムアウト値
・ER_BOO	ER または BOOL
・ER_ID	ER または ID
・ER_UINT	ER または UINT

■unsigned int型に定義しているデータ型

・UINT	符号無し整数
・ATR	属性
・STAT	状態
・MODE	動作モード
・RELTIM	相対時間
・TEXPTN	タスク例外要因のビットパターン
・FLGPTN	イベントフラグのビットパターン

■size_t型に定義しているデータ型

・SIZE	サイズ

静的APIとコンフィギュレータ

■コンフィギュレータ

- コンフィギュレーションファイルに静的APIを記述
- カーネル初期化ファイル“kernel_cfg.c”とID自動割付けファイル“kernel_id.h”などを生成

■静的API

- ”CRE_“で始まるAPI
- コンフィギュレーションファイルに記述
- 各IDが自動で生成されるために、コーディングミスによる重複がなくなる

■記述例

```
CRE_TSK(TASK1, { TA_HLNG, (VP_INT) 1, task, MID_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(TASK2, { TA_HLNG, (VP_INT) 2, task, MID_PRIORITY, STACK_SIZE, NULL });
CRE_TSK(TASK3, { TA_HLNG, (VP_INT) 3, task, MID_PRIORITY, STACK_SIZE, NULL });
```

タスク管理機能

■タスク管理機能

- CRE_TSK　タスクの生成（静的API）
- act_tsk, iact_tsk　タスクの起動
- can_act　タスク起動要求のキャンセル
- ext_tsk　自タスクの終了
- ter_tsk　タスクの強制終了
- chg_pri　タスク優先度の変更
- get_pri　タスク優先度の参照

タスク付属同期機能

■タスク付属同期機能

•slp_tsk	起床待ち
•tslp_tsk	起床待ち（タイムアウトあり）
•wup_tsk, iwup_tsk	タスクの起床
•can_wup	タスク起床要求のキャンセル
•rel_wai, irel_wai	待ち状態の強制解除
•sus_tsk	強制待ち状態への移行
•rsm_tsk	強制待ち状態からの再開
•frsm_tsk	強制待ち状態からの強制再開
•dly_tsk	自タスクの遅延

タスク例外処理機能

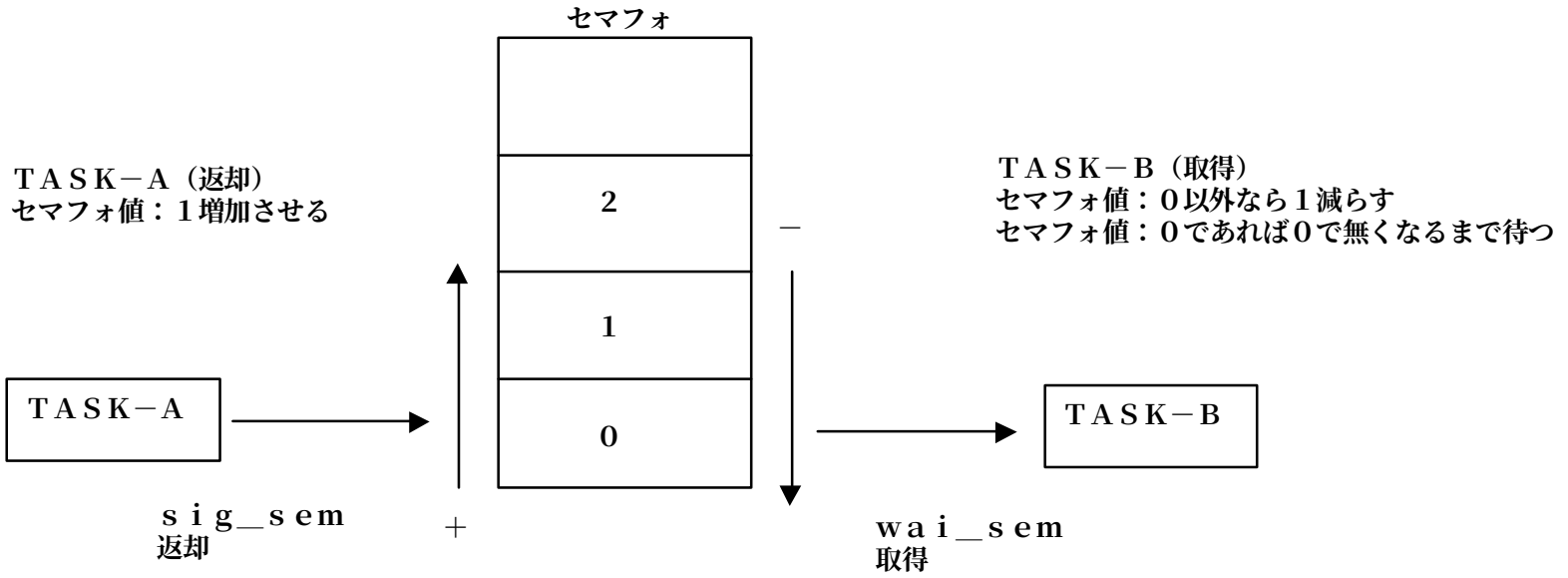
■ タスク例外処理機能

- DEF_TEX　　タスク例外処理ルーチンの定義（静的API）
- ras_tex, iras_tex　　タスク例外処理の要求
- dis_tex　　タスク例外処理の禁止
- ena_tex　　タスク例外処理の許可
- sns_tex　　タスク例外処理禁止状態の参照

セマフォ機能

■セマフォ

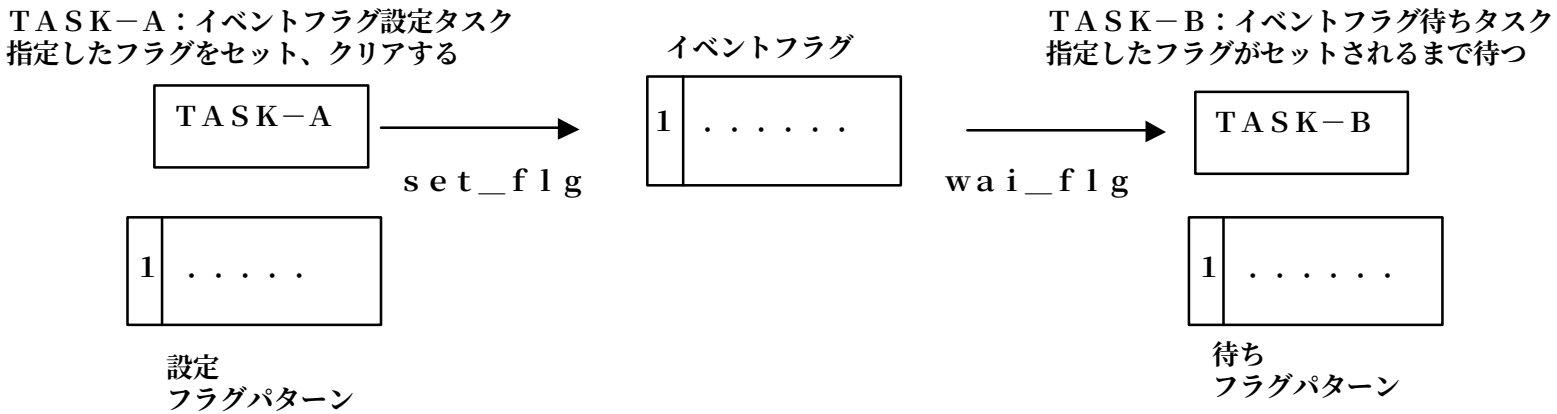
- CRE_SEM　　セマフォの生成（静的API）
- sig_sem, isig_sem　　セマフォ資源の返却
- wai_sem　　セマフォ資源の獲得
- pol_sem　　セマフォ資源の獲得（ポーリング）
- twai_sem　　セマフォ資源の獲得（タイムアウトあり）



イベントフラグ機能

■イベントフラグ

- CRE_FLG　　イベントフラグの生成（静的API）
- set_flg, iset_flg　　イベントフラグのセット
- clr_flg　　イベントフラグのクリア
- wai_flg　　イベントフラグ待ち
- pol_flg　　イベントフラグ待ち（ポーリング）
- twai_flg　　イベントフラグ待ち（タイムアウトあり）



データキュー機能

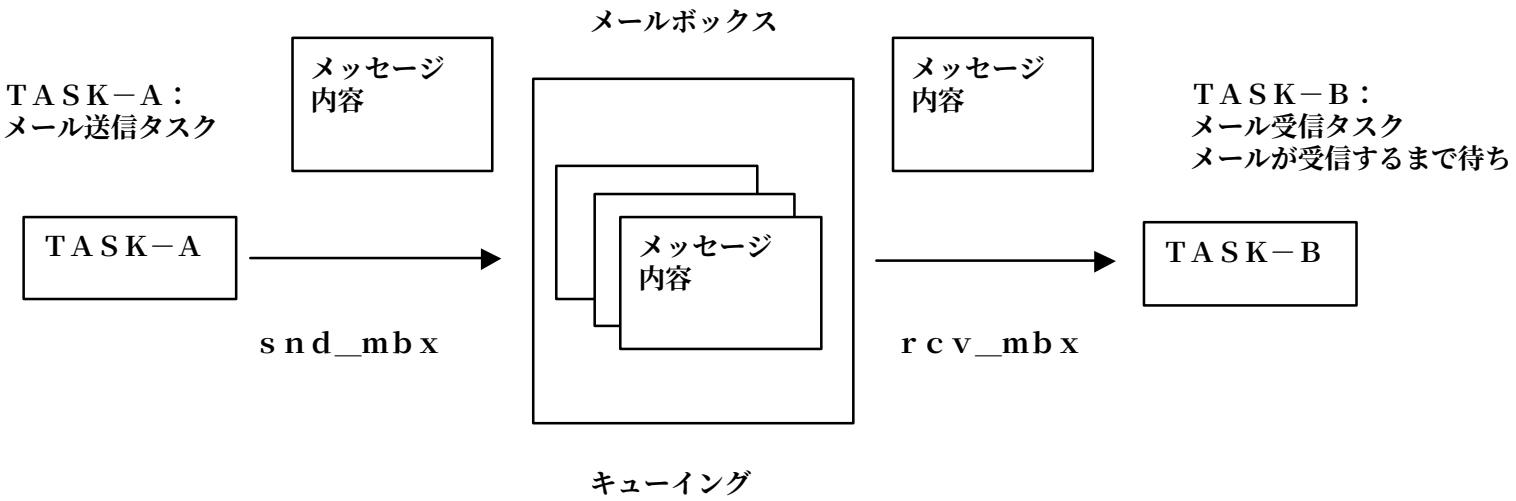
■データキュー

- CRE_DTQ　　データキューの生成（静的API）
- snd_dtq　　データキューへの送信
- psnd_dtq, ipsnd_dtq　　データキューへの送信（ポーリング）
- tsnd_dtq　　データキューへの送信（タイムアウトあり）
- fsnd_dtq, ifsnd_dtq　　データキューへの強制送信
- rcv_dtq　　データキューからの受信
- prcv_dtq　　データキューからの受信（ポーリング）
- trcv_dtq　　データキューからの受信（タイムアウトあり）

メールボックス機能

■メールボックス

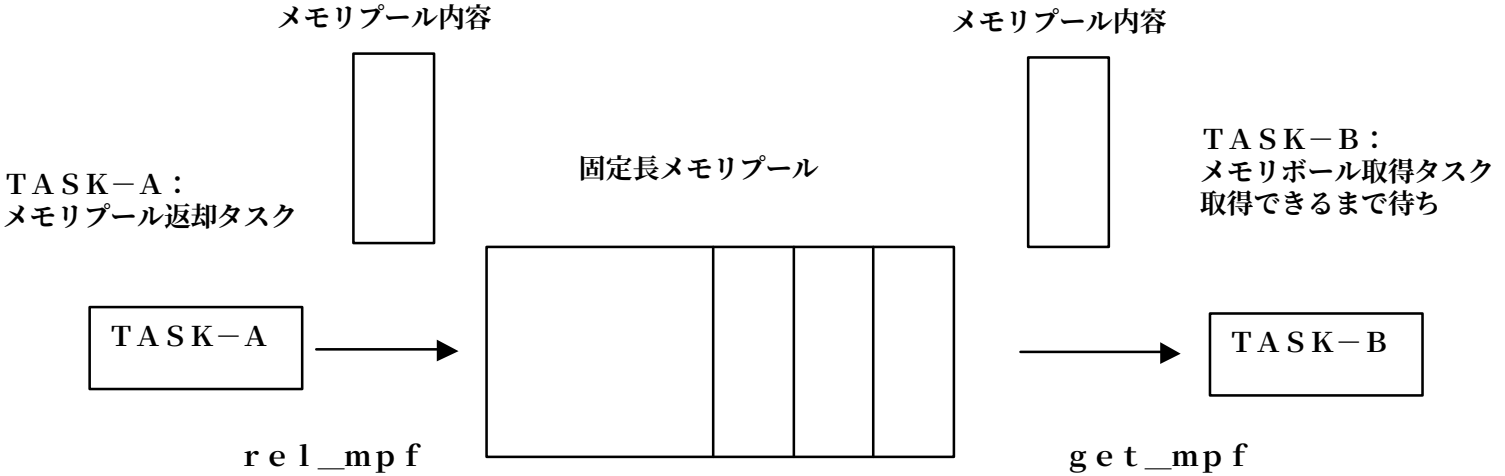
- CRE_MBX　　メールボックスの生成（静的API）
- snd_mbx　　メールボックスへの送信
- rcv_mbx　　メールボックスからの受信
- prcv_mbx　　メールボックスからの受信（ポーリング）
- trcv_mbx　　メールボックスからの受信（タイムアウトあり）



メモリプール機能

■固定長メモリプール

- CRE_MPF　　固定長メモリプールの生成（静的API）
- get_mpf　　固定長メモリブロックの獲得
- pget_mpf　　固定長メモリブロックの獲得（ポーリング）
- tget_mpf　　固定長メモリブロックの獲得（タイムアウトあり）
- rel_mpf　　固定長メモリブロックの返却



システム時刻管理機能

■システム時刻管理

- set_tim　　システム時刻の設定
- get_tim　　システム時刻の参照
- isig_tim　　タイムティックの供給

周期ハンドラ機能

■周期ハンドラ

- CRE_CYC　　周期ハンドラの生成（静的API）
- sta_cyc　　周期ハンドラの動作開始
- stp_cyc　　周期ハンドラの動作停止

システム管理機能

■システム状態管理機能

- rot_rdq, irot_rdq　タスクの優先順位の回転
- get_tid, iget_tid　実行状態のタスクIDの参照
- loc_cpu, iloc_cpu　CPUロック状態への移行
- unl_cpu, iunl_cpu　CPUロック状態の解除
- dis_dsp　ディスパッチの禁止
- ena_dsp　ディスパッチの許可
- sns_ctx　コンテキストの参照
- sns_loc　CPUロック状態の参照
- sns_dsp　ディスパッチ禁止状態の参照
- sns_dpn　ディスパッチ保留状態の参照
- vsns_ini　カーネル動作状態の参照

割込み管理機能

■割込み管理機能

- DEF_INH　　割込みハンドラの定義（静的API）
- dis_int　　割込みの禁止
- ena_int　　割込みの許可
- chg_ixx　　割込みマスクの変更
- get_ixx　　割込みマスクの参照

システム構成管理機能

■システム構成管理機能

- DEF_EXC　　CPU例外ハンドラの定義（静的API）
- ATT_INI　　初期化ルーチンの追加（静的API）
- VATT_TER　　終了処理ルーチンの追加（静的API）

TOPPERS/JSPカーネルによるプログラミング :1

■タスク分割の考え方

- ●タスク
  - ・プログラムが並行で実行される単位
  - ・プロセッサを仮想的に多重化
- ●仕様変更に備える
- ●堅牢性 → メモリ保護機能の利用 → リアルタイム性が損なわれる?
- ●構造化/モジュール化
- ●カーネルの選択
  - ・JSPかFI4か
- ●APIの選択
  - ・同期
  - ・排他制御
  - ・メッセージ通信. . .

TOPPERS/JSPカーネルによるプログラミング :2

■コンフィギュレータの役割

- コンフィギュレーションファイルに静的APIを記述
- カーネル初期化ファイル “kernel_cfg.c ”とID自動割付けファイル “kernel_id.h ” などを生成

■コンフィギュレータの利点

- ID自動割付によりミスが少ないコーディング
- ソフトウェア部品変更/追加への対応
- CPU変更への対応
- 基板変更への対応
- OSベンダ変更への対応