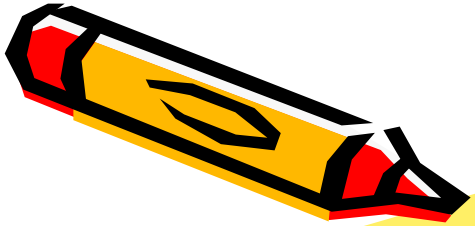


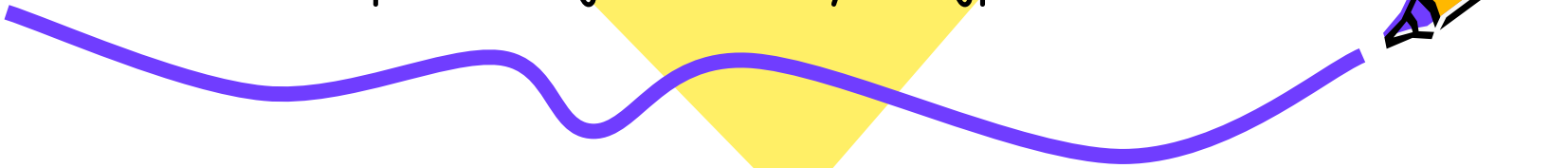
ソフトウェア第三
―実世界とインタフェースするソフトウェアシステム―

http://www.jsk.t.u-tokyo.ac.jp/~ikuo/lec/soft3/

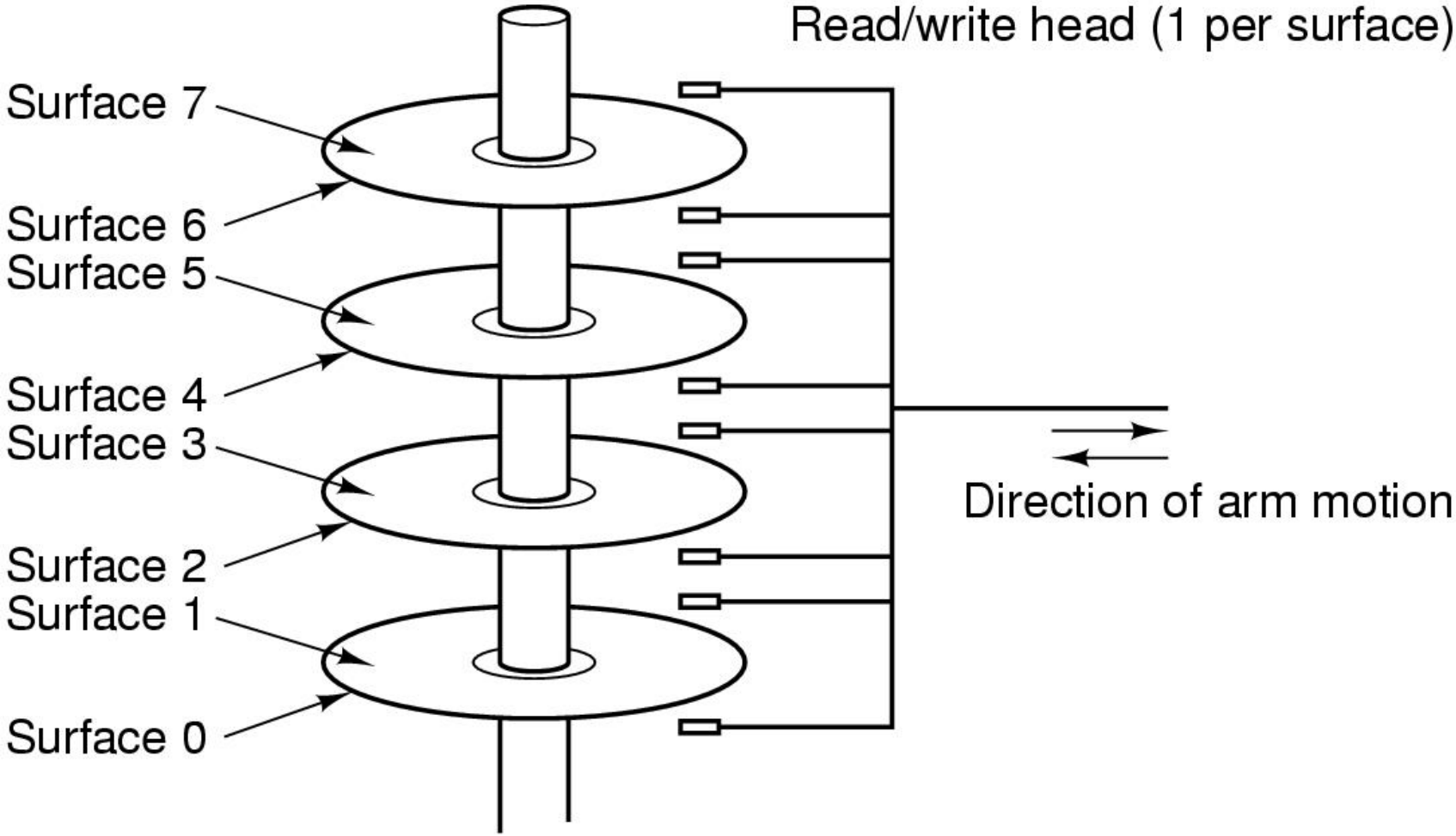
機械情報工学科　水内郁夫

2008年11月17日

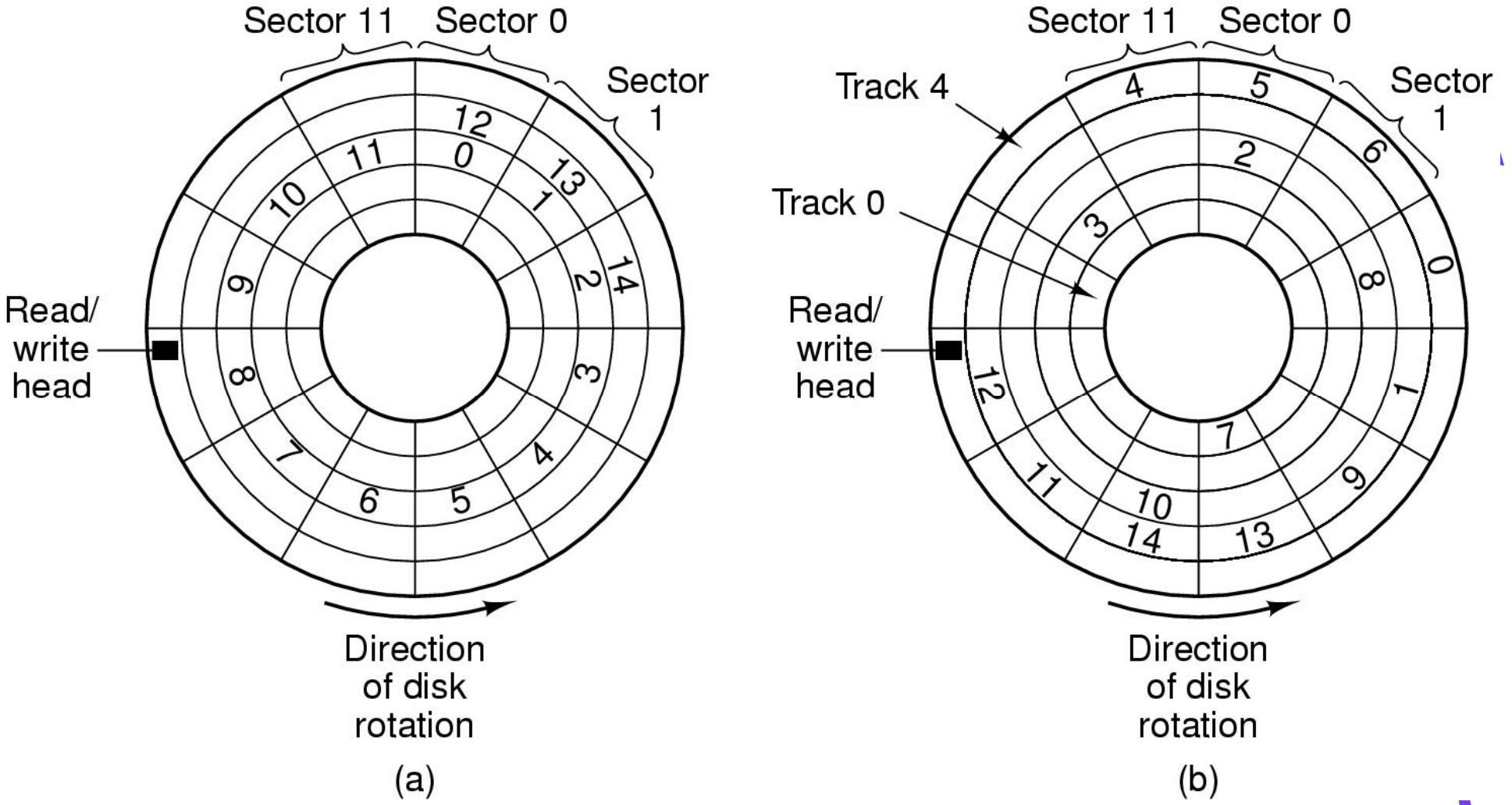
http://www.jsk.t.u-tokyo.ac.jp/~ikuo/



4枚のプラッタを持つHDD

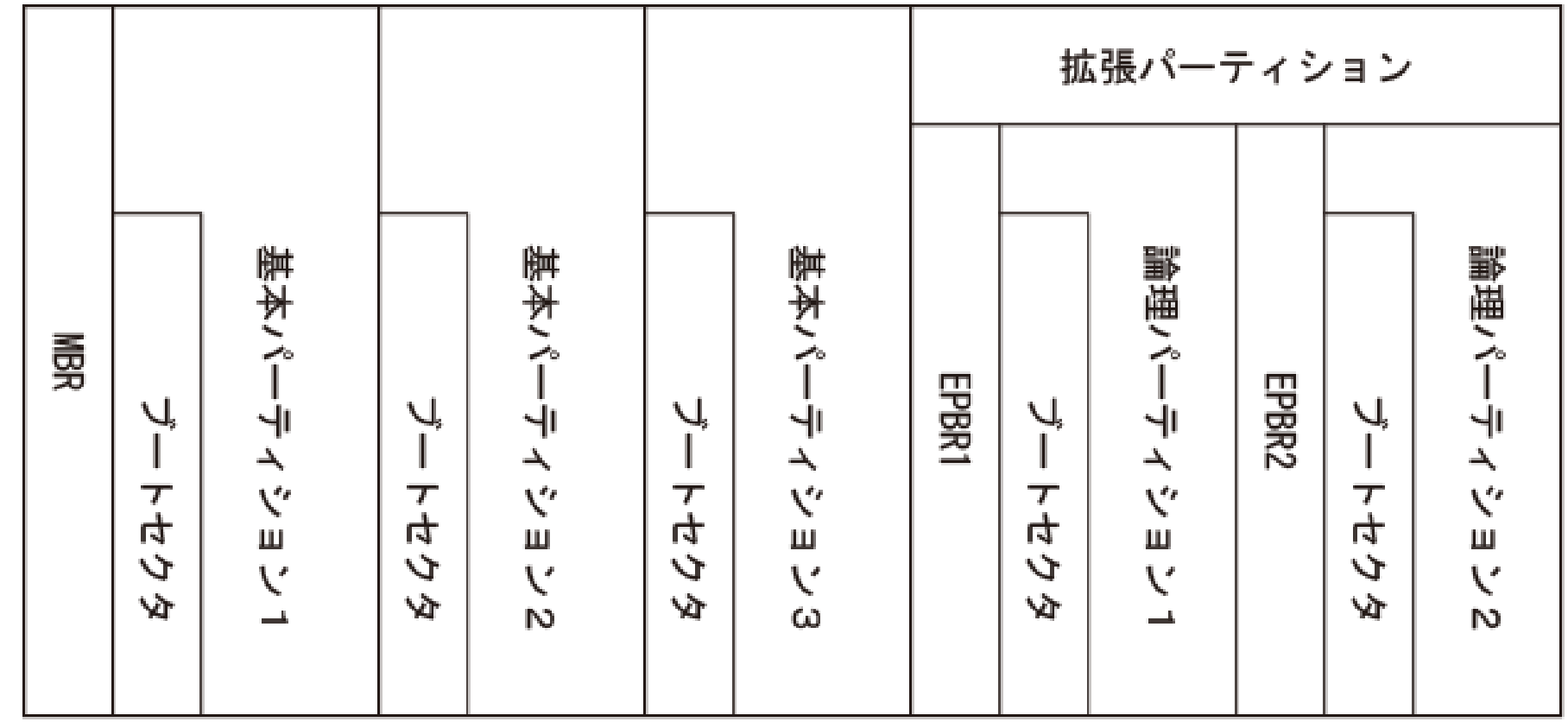


1枚のディスクの片面、CHS

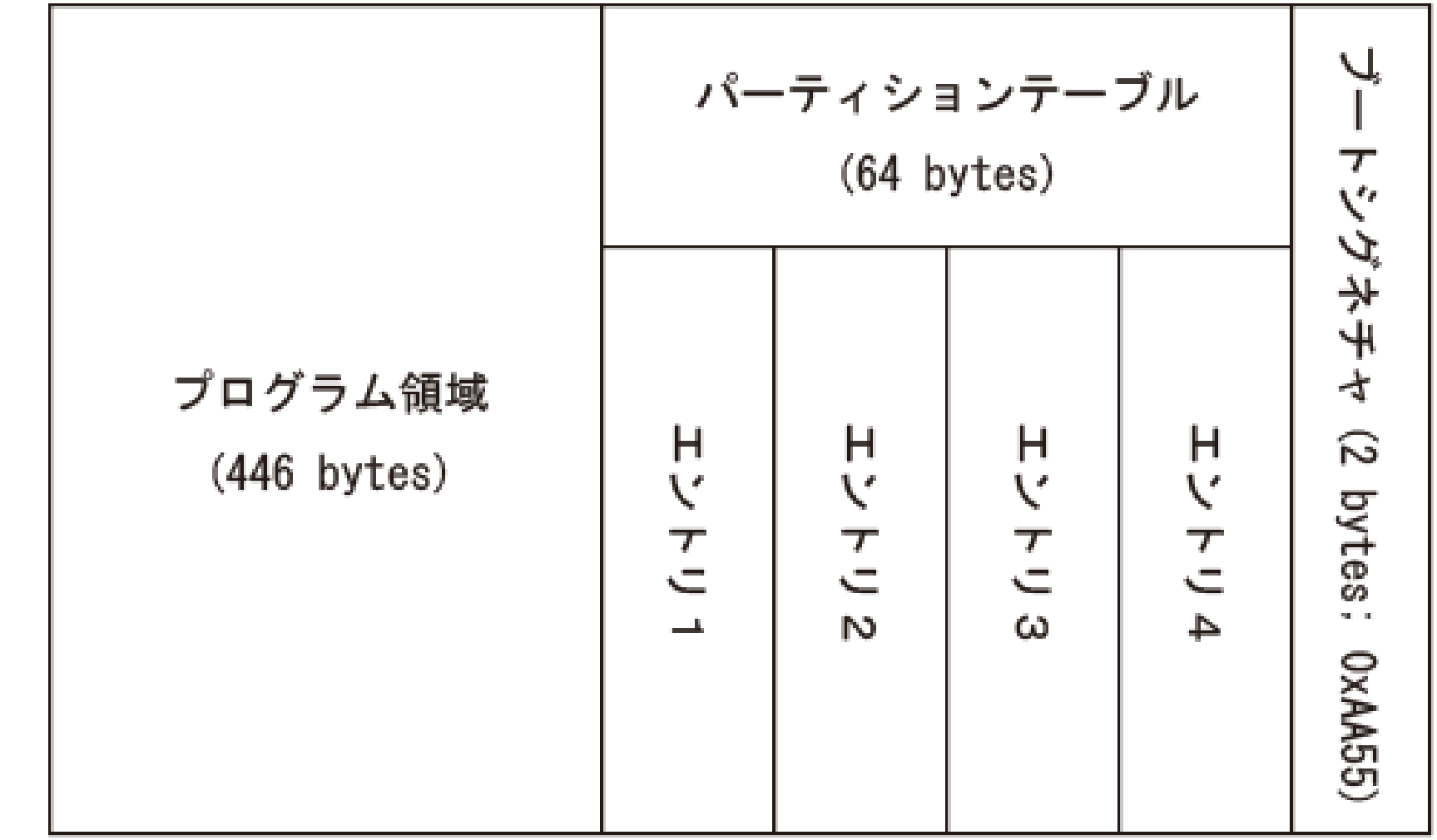


パーティション

- primary partition
- extended partition
- logical partition



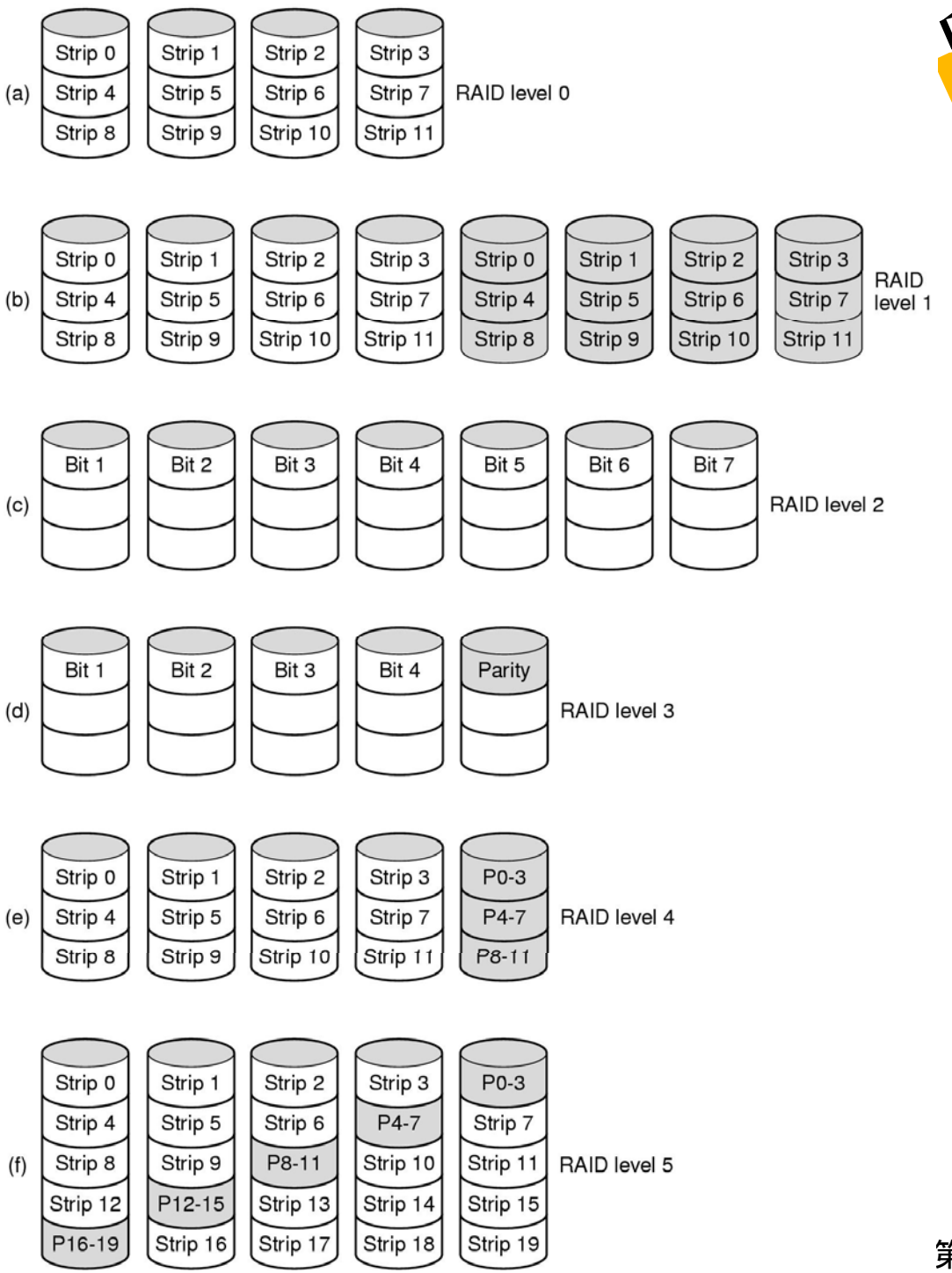
マスターブートレコード（MBR）



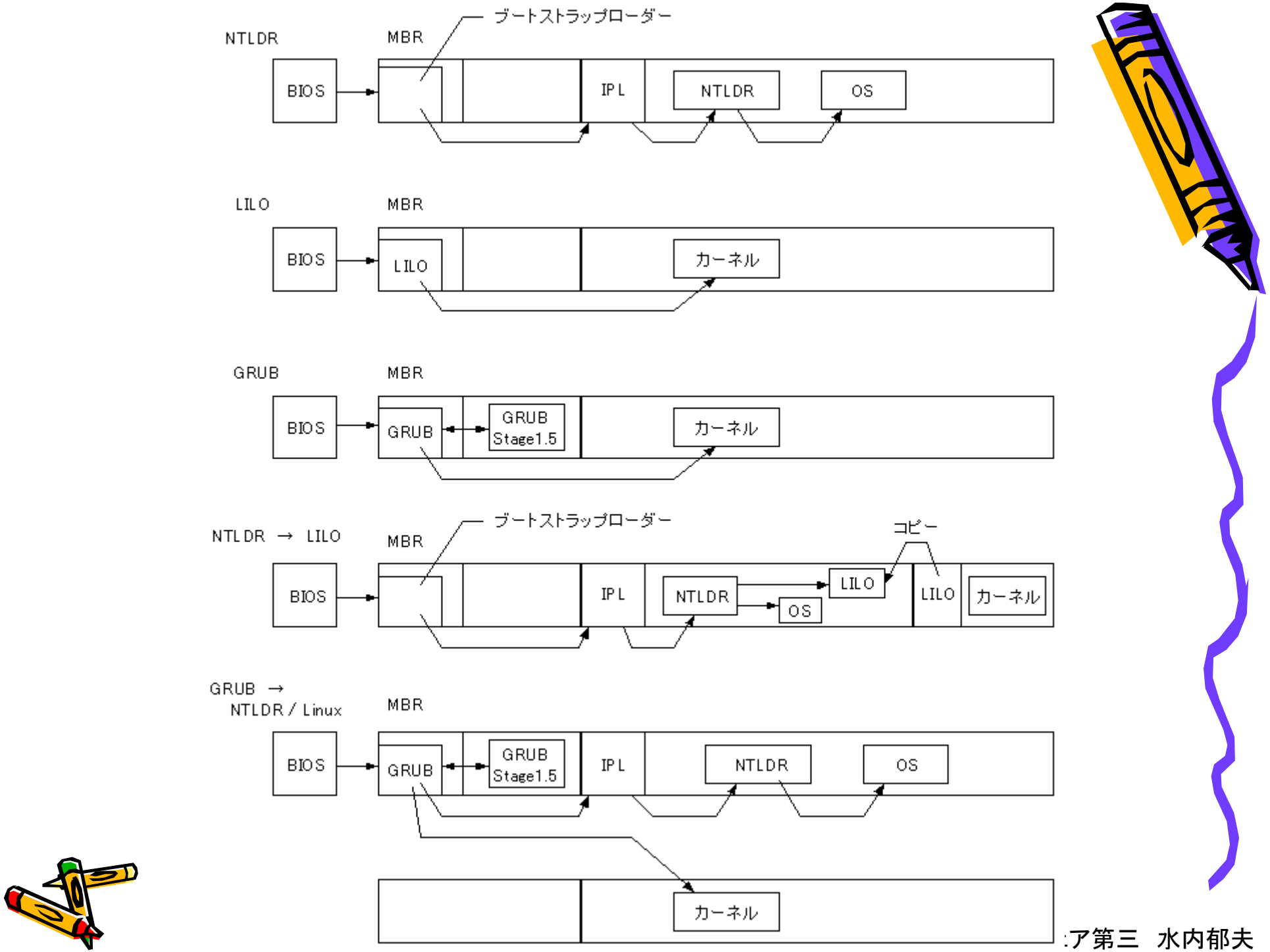
RAID

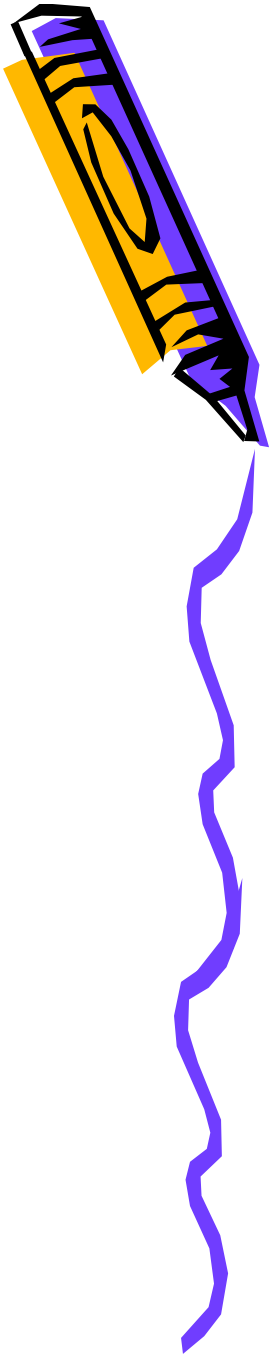
- D.Patterson
- UCB
- RISCもこの人

- パタヘネ本
- ヘネパタ本



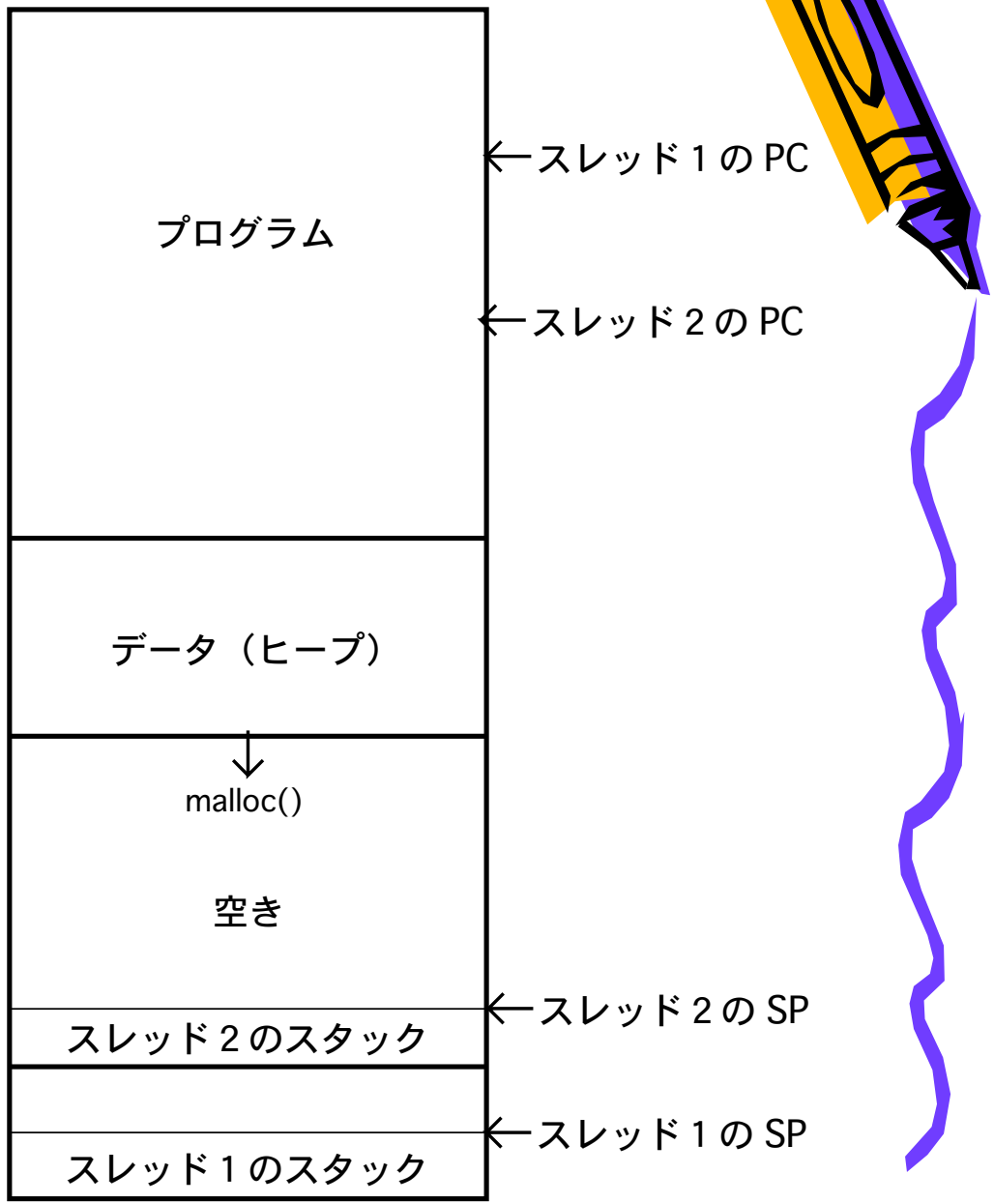
第三　水内郁夫

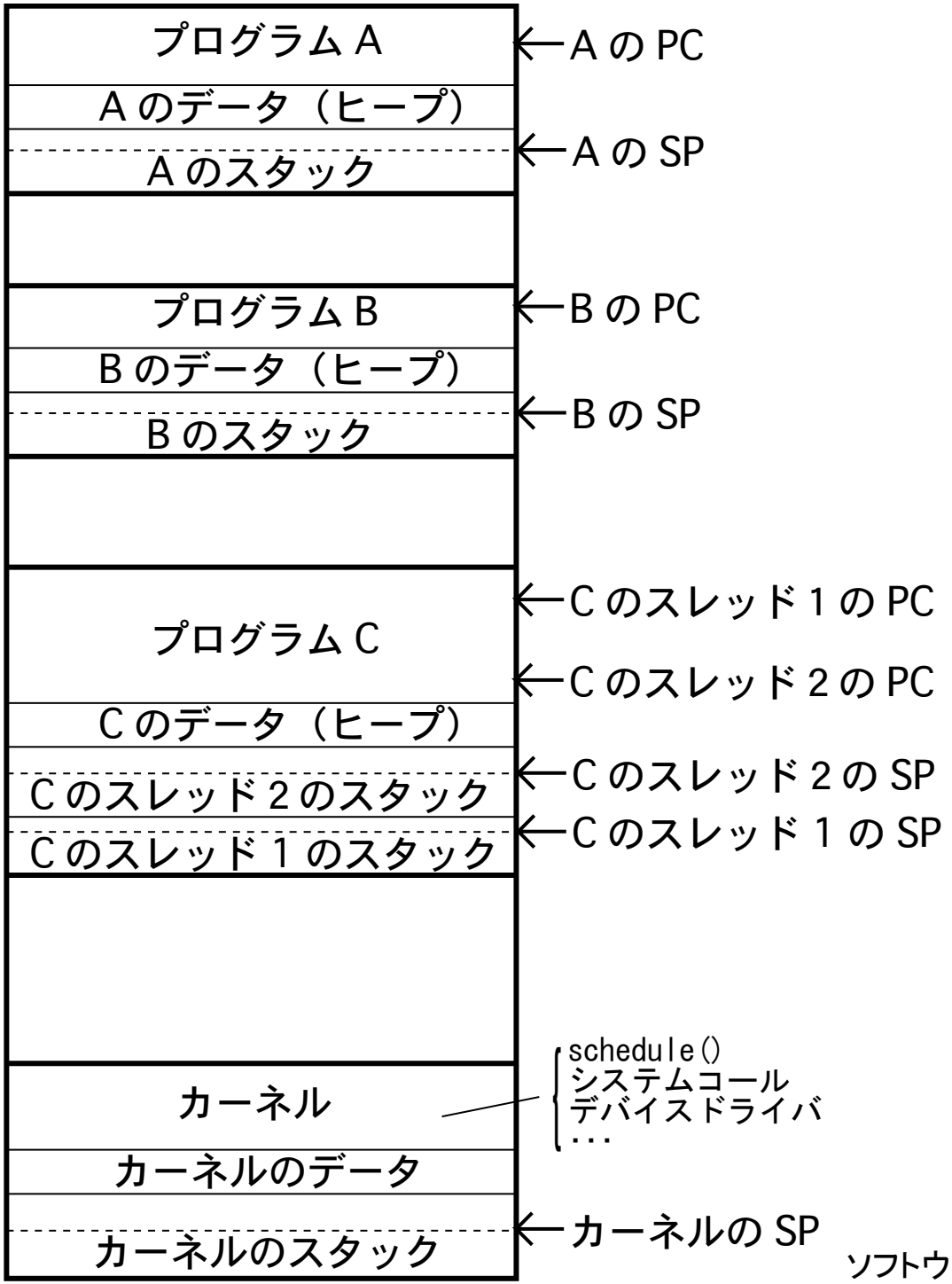

NTLDR
MBR
ブートストラップローダー
BIOS
IPL
NTLDR
OS
LILO
MBR
BIOS
LILO
カーネル
GRUB
MBR
BIOS
GRUB
GRUB
Stage1.5
カーネル
NTLDR → LILO
MBR
ブートストラップローダー
コピー
BIOS
IPL
NTLDR
LILO
OS
LILO
カーネル
GRUB →
NTLDR / Linux
MBR
BIOS
GRUB
GRUB
Stage1.5
IPL
NTLDR
OS
カーネル

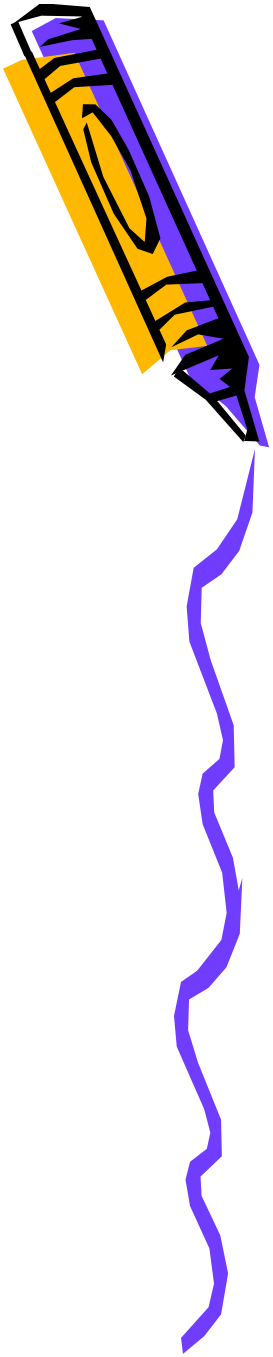


複数スレッドを持つプロセスのメモリイメージ

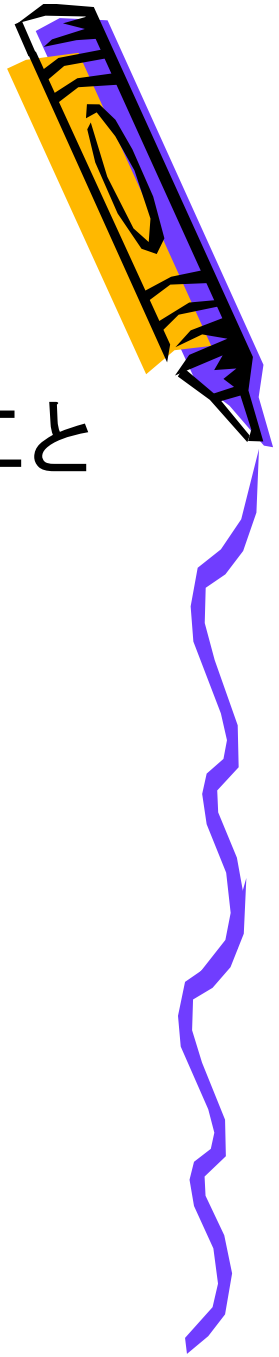
- PC、SP、レジスタは、スレッド毎に保持。
- データは共有（共有メモリ）




プログラム A
A のデータ（ヒープ）
A のスタック
プログラム B
B のデータ（ヒープ）
B のスタック
プログラム C
C のデータ（ヒープ）
C のスレッド 2 のスタック
C のスレッド 1 のスタック
カーネル
カーネルのデータ
カーネルのスタック
A の PC
A の SP
B の PC
B の SP
C のスレッド 1 の PC
C のスレッド 2 の PC
C のスレッド 2 の SP
C のスレッド 1 の SP
カーネルの SP
schedule()
システムコール
デバイスドライバ
・・・

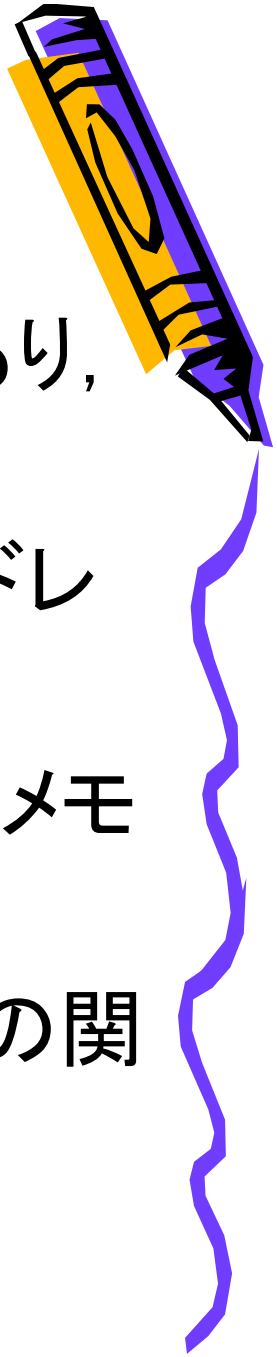


カーネルとは？



- OSの中枢部である，一つのプログラムのこと
- カーネル空間とユーザ空間
- root のみが実行可能
- ユーザプロセスからのシステムコール

カーネル空間と ユーザープロセス空間

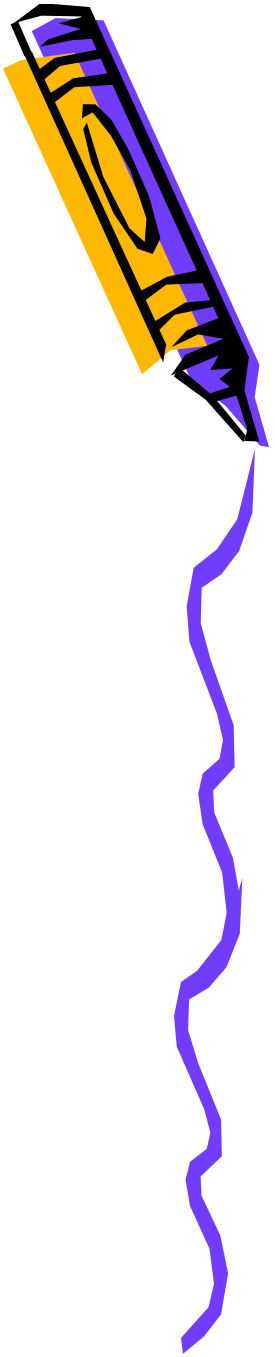


- カーネルのメモリ空間は物理アドレスであり，不動である．(ページアウトしない)
- ユーザープロセスのメモリ空間は論理アドレスで，ページングが発生する
- ユーザープロセスは簡単にはカーネルのメモリ空間にアクセスする事はできない
- メモリの状態をやりとりするために，専用の関数が必要

システムコール

- アプリケーション・プログラムがOSのサービスを呼び出す際の規約やメカニズム
- ユーザプロセスモードからカーネルモードに切り替わる
- システムコールの最後に，再スケジューリングが行われる

システムコールの例



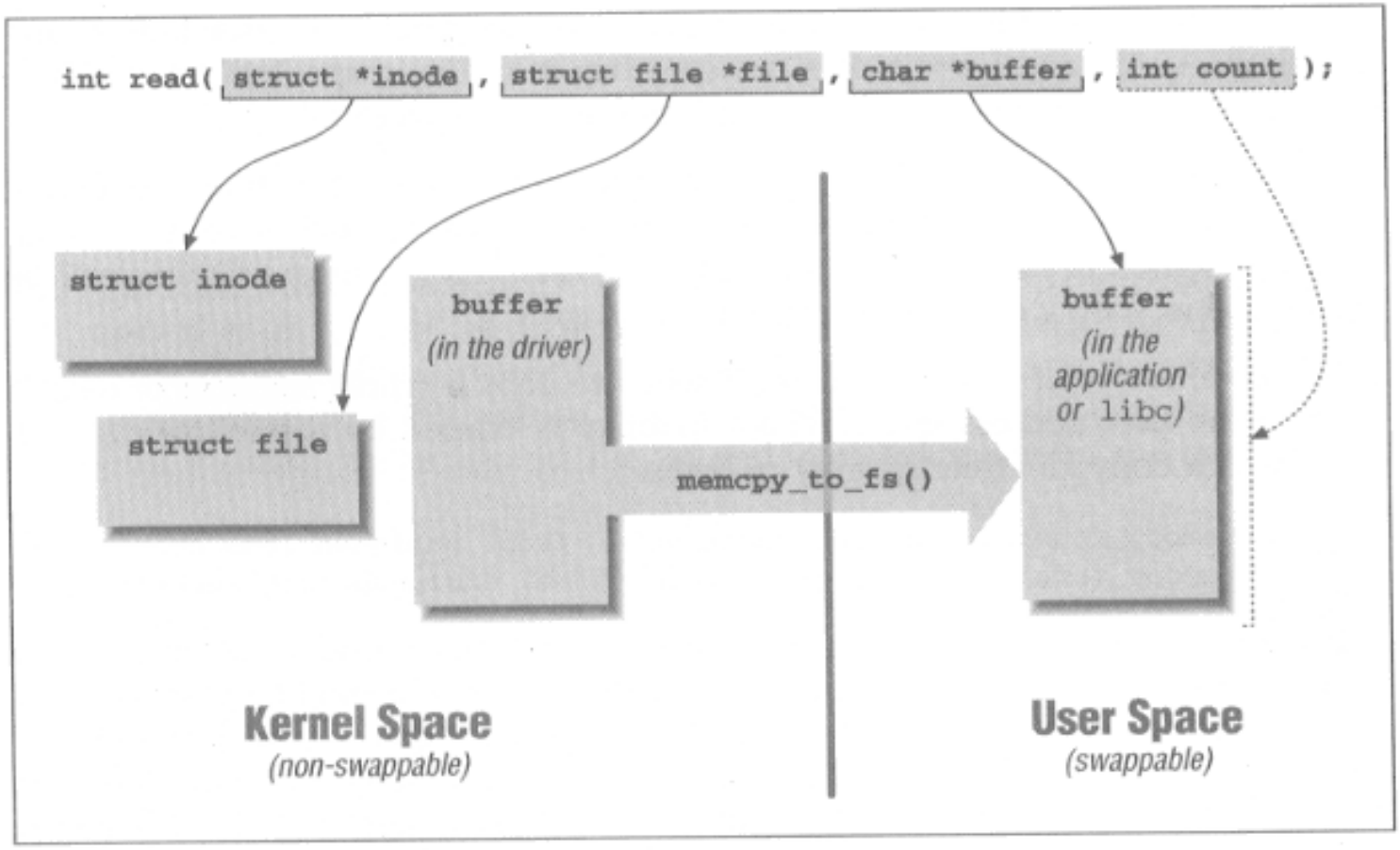
- プロセス管理
  - fork, exec, wait
- ファイル管理
  - open, read , write , close
- プロセス間通信関係
  - signal, kill, pipe
- デバイスドライバ関係
  - read, write, ioctl, open, close

デバイスドライバとは？

- ハードウェアと，ソフトウェアを結ぶためのインタフェース
- OSの起動時にメモリ上に読み込まれて常駐し、周辺装置からのハード的な割り込みによって駆動される
- ハードウェアの隠蔽化，カプセル化，抽象化

Read システムコールの中身

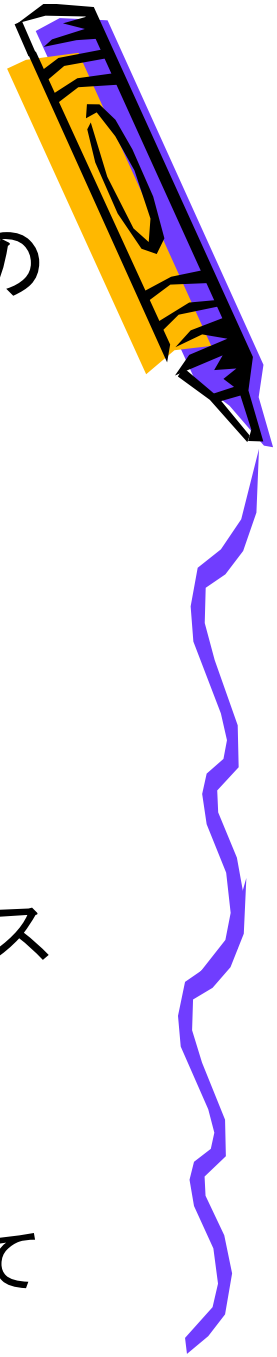
- copy_to_user()：ユーザ領域へのメモリ転送



Linuxのローダブルモジュール

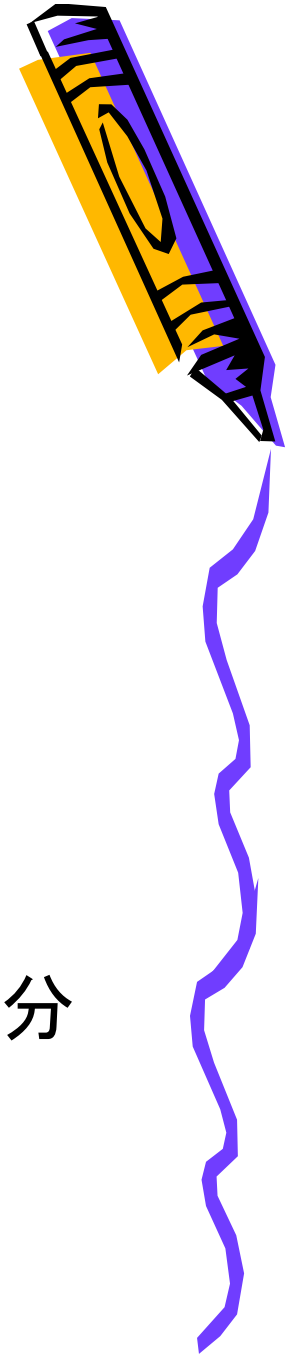
- デバイスドライバをモジュールにする。
- モジュールの利点
  - カーネルがコンパクトになる
  - カーネルの再構築の必要性が大幅に減り、新しいデバイスへの対応が容易になる
- モジュールの利用
  - insmod でモジュールのカーネルへの組込み
  - rmmod でカーネルからの削除

デバイスファイル



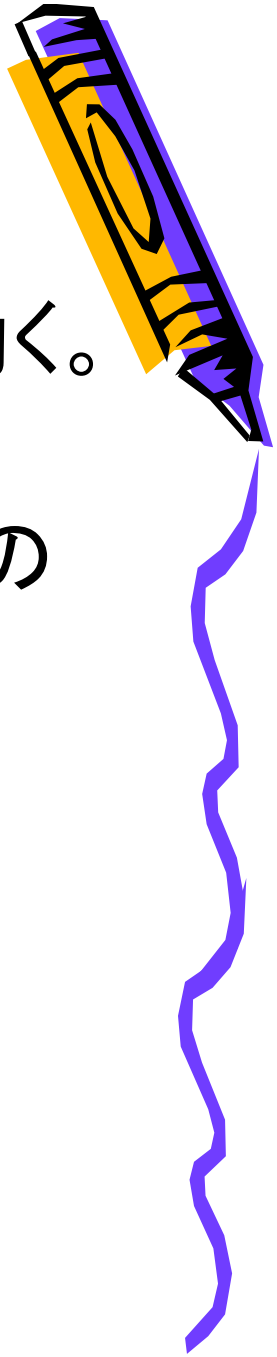
- デバイスドライバ(カーネル部)とのデータのやりとりの窓口
  - /dev/audio, /dev/usb/mouse0
  - /dev/hda, /dev/hda0
- read, write, ioctl の対象
- メジャー番号(Linux)
  - 1つのデバドラ(モジュール)で管轄するデバイスは統一して付ける.
- マイナー番号(Linux)
  - そのデバドラ内部で区別するデバイスに対して個別に割り振る

Linuxデバイス・ドライバの構造



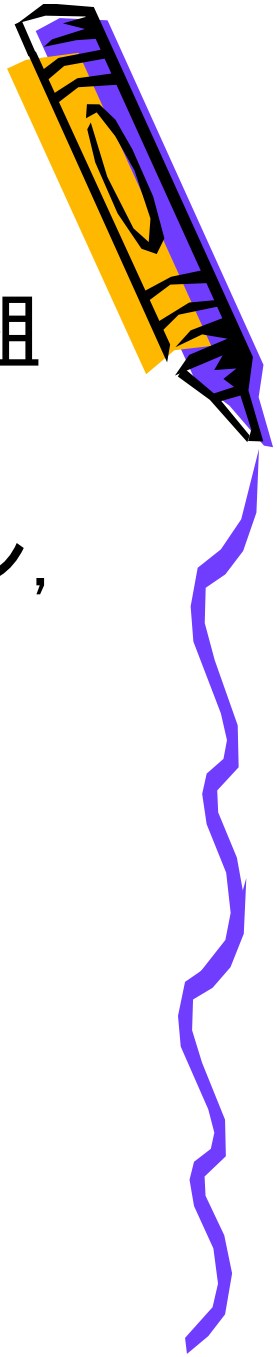
- 初期化ルーチン
  - デバドラが組み込まれる瞬間に呼ばれる
  - 関数名は init_module
- トップハーフルーチン
  - 上位から(間接的に)コールされる部分
  - read, open などに相当
- ボトムハーフルーチン
  - ハードウェア割り込みによって駆動される部分
  - interrupt 系

デバドラの制約条件

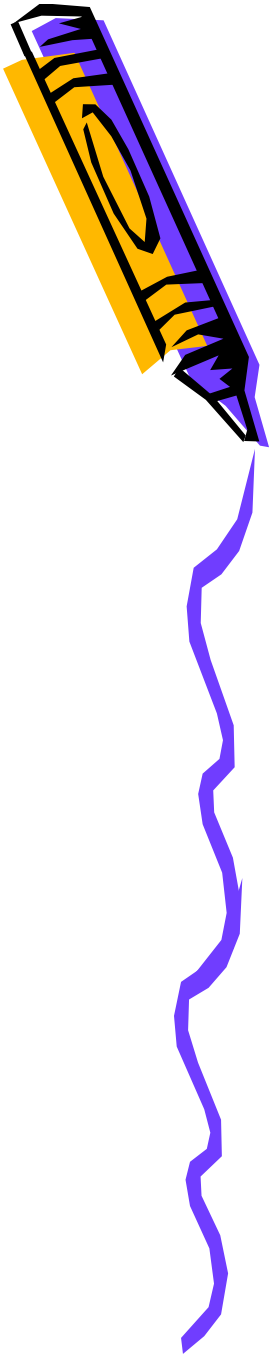


- デバイスドライバ内部は、カーネル空間で動く。
- デバイスドライバ内部では、OSが提供するreadなどのシステムコール や printf などのライブラリ群はいっさい使えない.
- malloc の代わりに kmalloc
  - しかもページアウトしない
- printf の代わりに printk
  - /var/log/syslogにも出力される。

init_module(void)

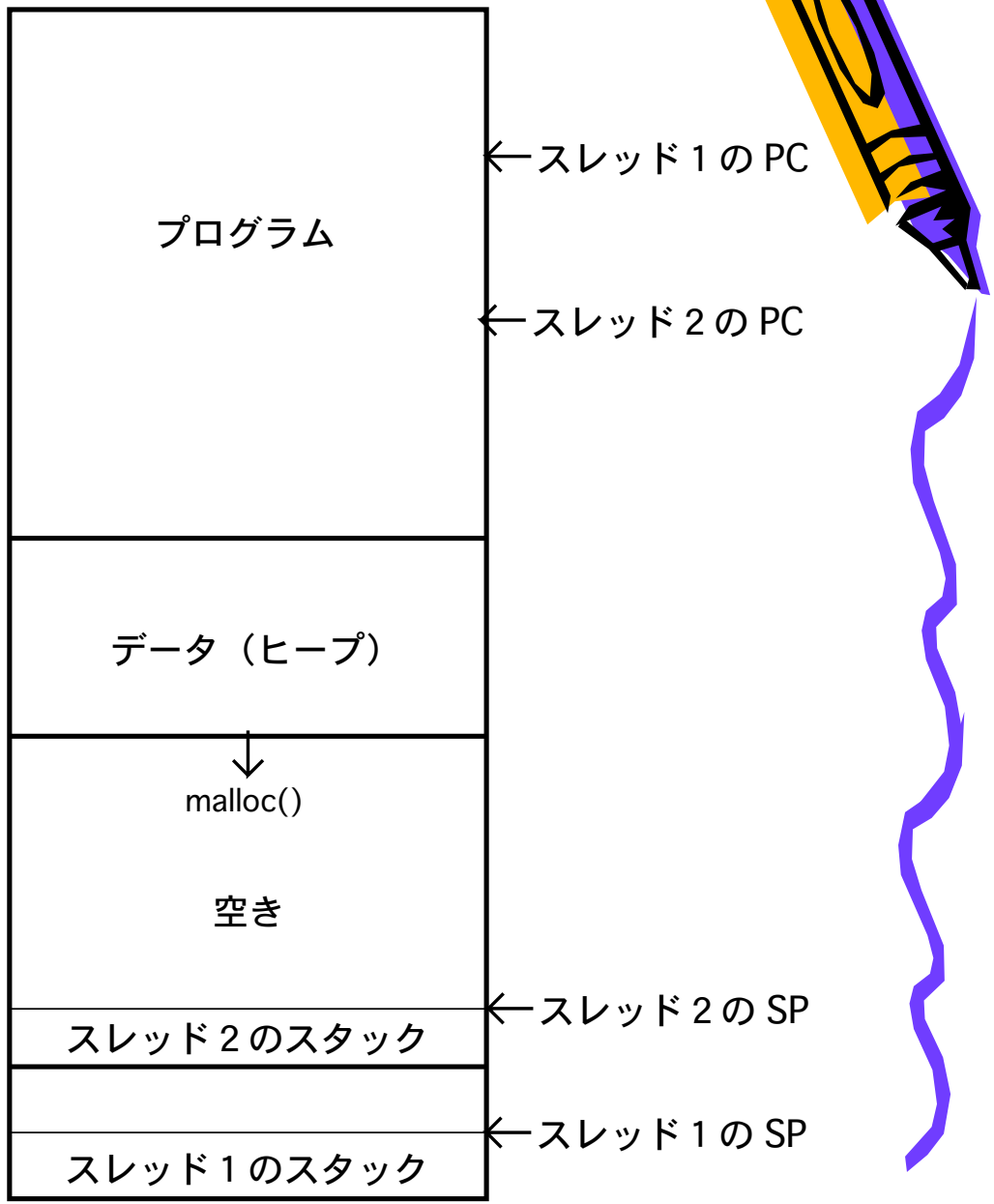


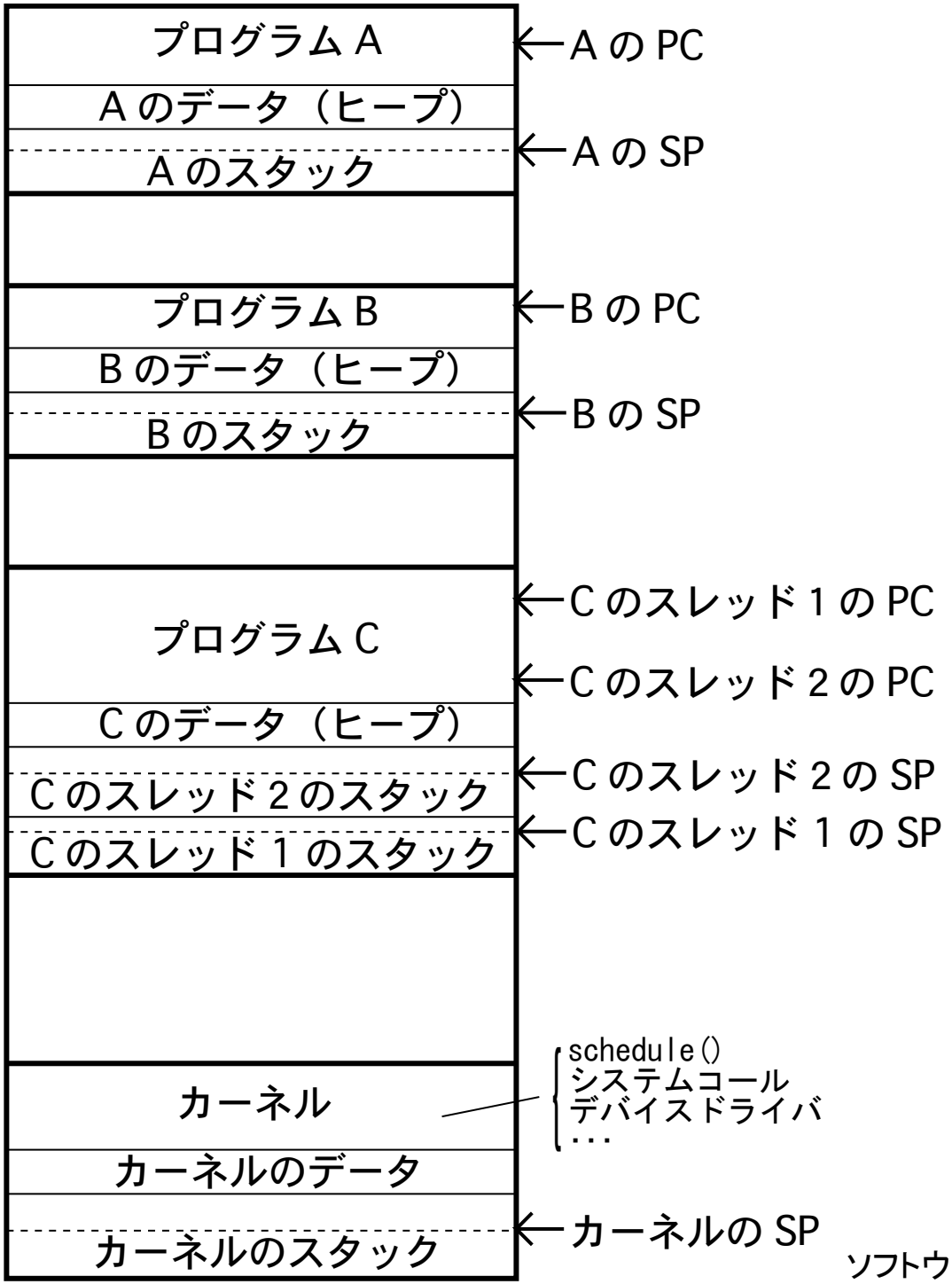
- モジュール(Linuxのデバイスドライバ)が組み込まれた時 (insmod の時)に呼ばれる
- register_chrdev でトップハーフルーチン，ボトムハーフルーチンを登録
- 反対にモジュールが削除される場合には，cleanup_module が呼ばれる

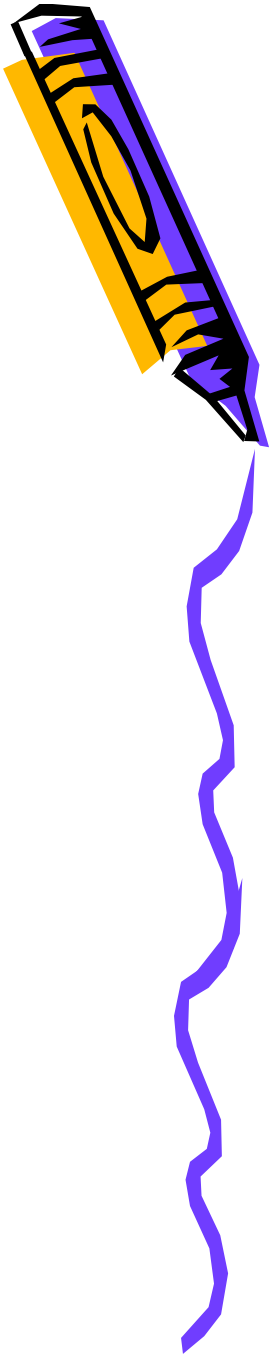


複数スレッドを持つプロセスのメモリイメージ

- PC、SP、レジスタは、スレッド毎に保持。
- データは共有（共有メモリ）



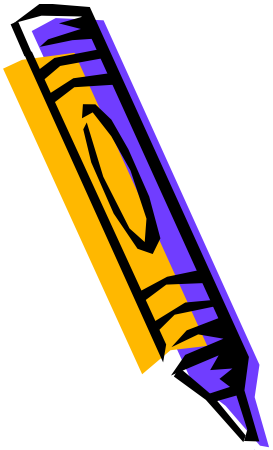

プログラム A
A のデータ（ヒープ）
A のスタック
プログラム B
B のデータ（ヒープ）
B のスタック
プログラム C
C のデータ（ヒープ）
C のスレッド 2 のスタック
C のスレッド 1 のスタック
カーネル
カーネルのデータ
カーネルのスタック
A の PC
A の SP
B の PC
B の SP
C のスレッド 1 の PC
C のスレッド 2 の PC
C のスレッド 2 の SP
C のスレッド 1 の SP
カーネルの SP
schedule()
システムコール
デバイスドライバ
・・・



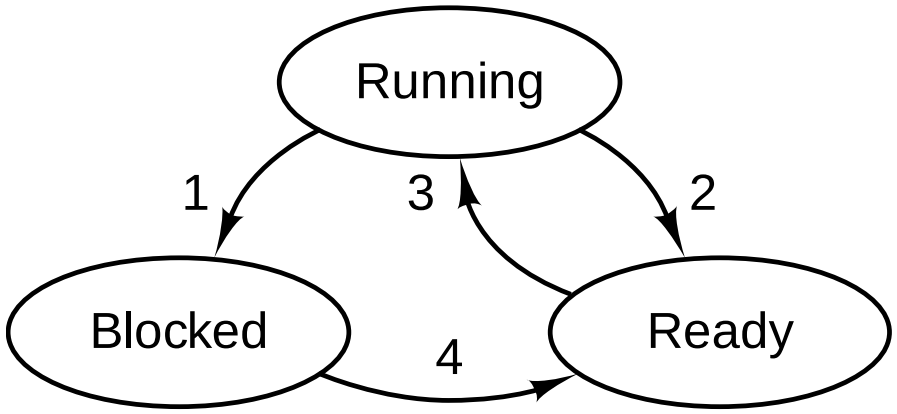
コンテキストスイッチ

- プロセス、スレッドの切替
- 実行状態を保存して、別のプロセス・スレッドの実行状態を読み出す。
- 切替時の作業
  - PC、SP、レジスタの、保存・読み出し
  - MMUの設定（仮想メモリと物理メモリの対応、TLB）
  - キャッシュのフラッシュ
- スレッドはプロセスより切替時の作業量は少ない。
- OSのスケジューラがコンテキストスイッチを行う。

プロセス・スレッドの三状態

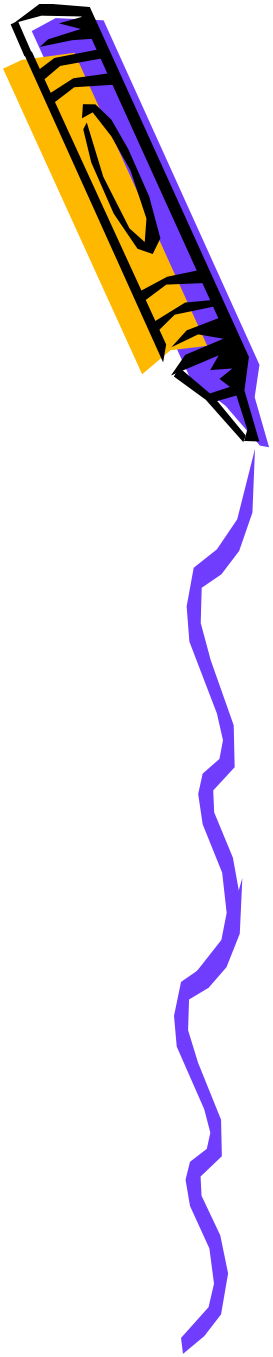


- 実行中: running
  - CPUで実行中
- 実行可能: ready
  - スケジューラからCPUを割り当てられるのを待っている。
- 待ち: blocked
  - システムコールの戻り待ち（IO待ち）等で、CPUを使う作業が無い状態。



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

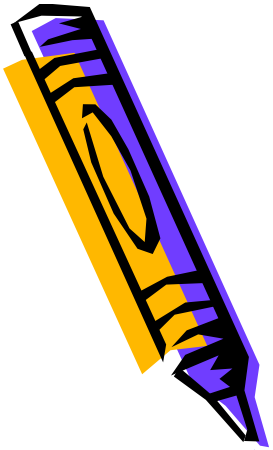
スケジューリング



- 実行可能状態のプロセス達のうち、どれを実行状態にするかを選ぶ。ディスパッチ。
- OSのスケジューラが呼ばれるタイミング
  - システムコールで待ち状態になった時
    - IO読み書き（read, write）
    - セマフォ待ち
    - スリープ
  - タイマ割り込み
  - ページフォールト時（トラップ）
- スケジュール時期
  - コンテキストスイッチのオーバヘッド
  - プリエンプション（preemption）

⇒リアルタイムシステム

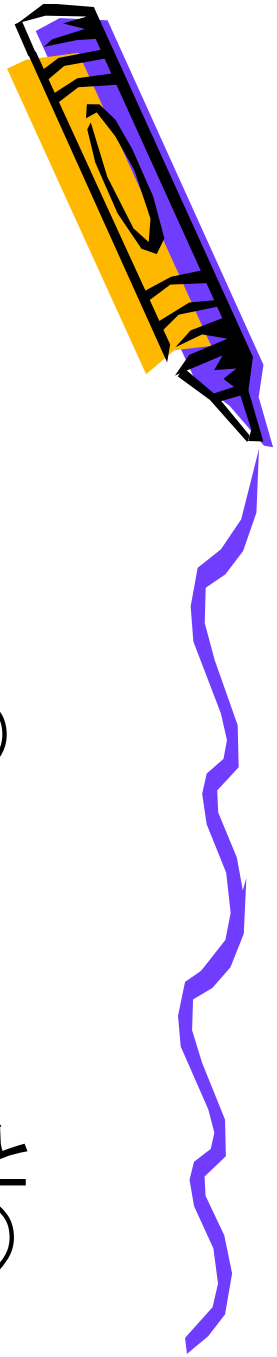
スケジューリングの例



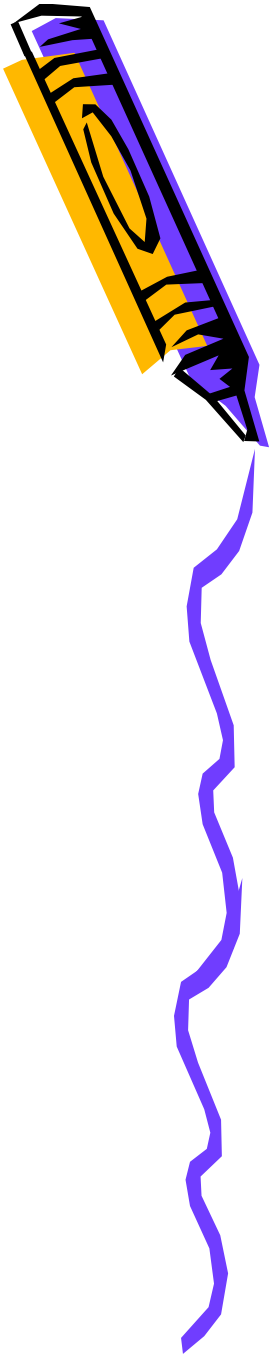
- schedule() 関数．プロセスから見れば，何もしないように見える

```
schedule()  {
    タスクキューの実行
    ボトムハーフハンドラ呼び出し          // bottom half handler
    プリエンプション要求をクリア
    if(スケジューラを呼び出したプロセスの状態がTASK_RUNNINGでないなら) {
        プロセスをRUNキューから外す
    }
    while (RUNキューに継っている全てのプロセスに対して)  {
        最も高いプライオリティのプロセスを探す
    }
    while (システム上の全てのプロセスに対して)   {
        プライオリティ再計算
    }
    プロセスコンテキストの切替え
}
```

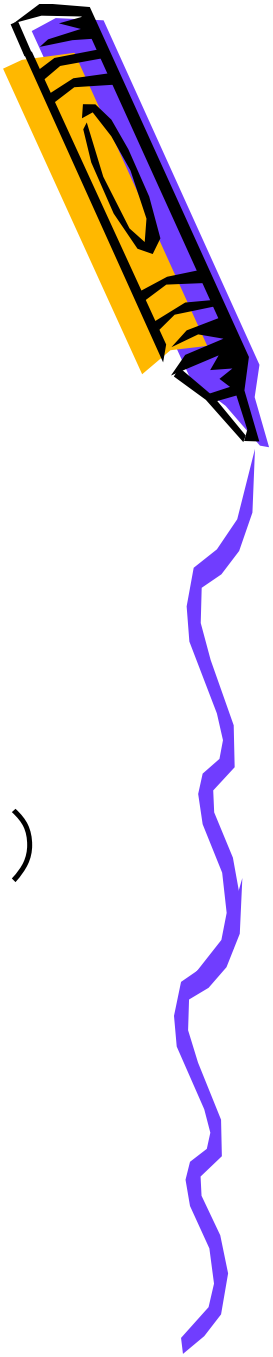
スケジュールの目標



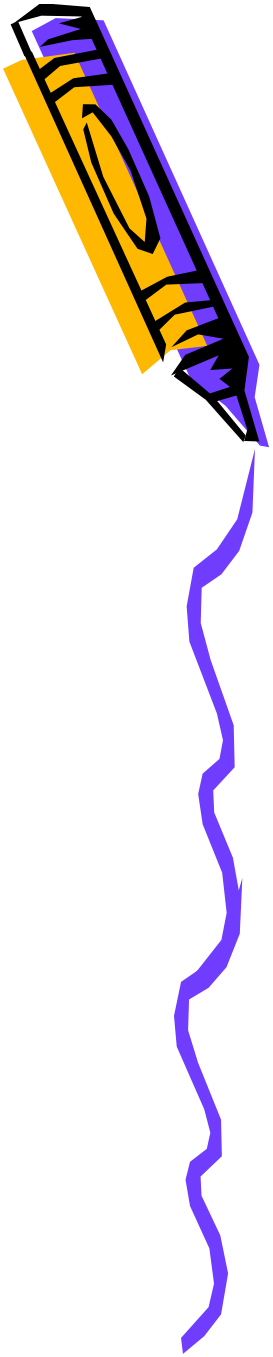
- スループット
  - 時間当たりの完了ジョブを最大にする。
- ターンアラウンド時間（レイテンシ）
  - ジョブ投入から完了までの時間を短くする。
- 対話応答時間
  - インタラクティブなタスク（コマンドシェルなど）
- CPU使用率
  - できるだけCPUのアイドル時間を減らす。
- デッドライン
  - リアルタイムシステム：時間内に完了しないと無価値（ハードRT）or価値が低下（ソフトRT）



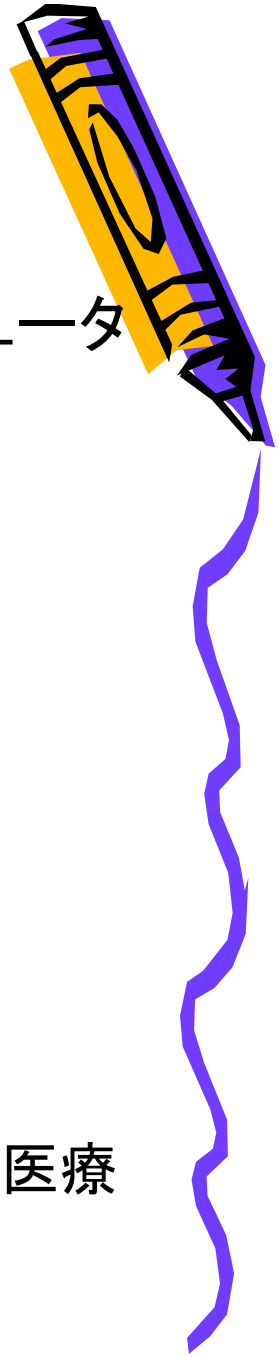
実世界とのインタフェース



- 外界からの入力（センサ）
  - AD変換器
  - スイッチ
  - チャタリング除去、フィルタ
  - アナログマルチプレクサ
  - パルス（矩形波）
- 外界への出力（アクチュエータ、エフェクタ）
  - DA変換器
  - デジタルIO
  - ドライバ・アンプ
  - PWM

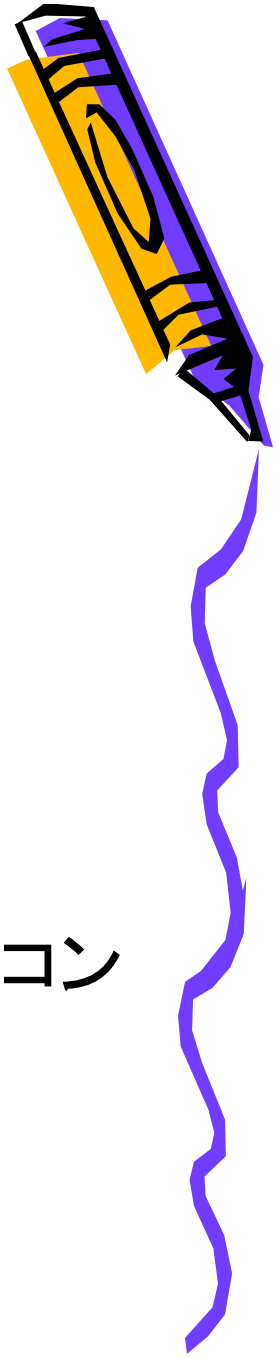


組み込みシステム



- 組み込みシステム(embedded system)
  - 各種の機器に組み込まれてその制御を行うコンピュータシステム（曖昧な定義だが）
- 例えば？
  - **AV機器**：TV、ビデオ、デジカメ、STB、オーディオ機器
  - **家電**：洗濯機、冷蔵庫、電子レンジ、炊飯器、エアコン
  - **情報・娯楽**：PDA、カーナビ、ゲーム、電子楽器
  - **PC周辺機器**：プリンタ、スキャナ、CD-ROMドライブ
  - **OA機器**：コピー、FAX
  - **通信機器**：携帯電話、人工衛星
  - **運輸機器**：自動車、列車、航空機などの制御機器
  - **工業、その他**：プラント制御、エレベータ、自動販売機、医療機器、ロボット制御

組み込みシステムの特徴

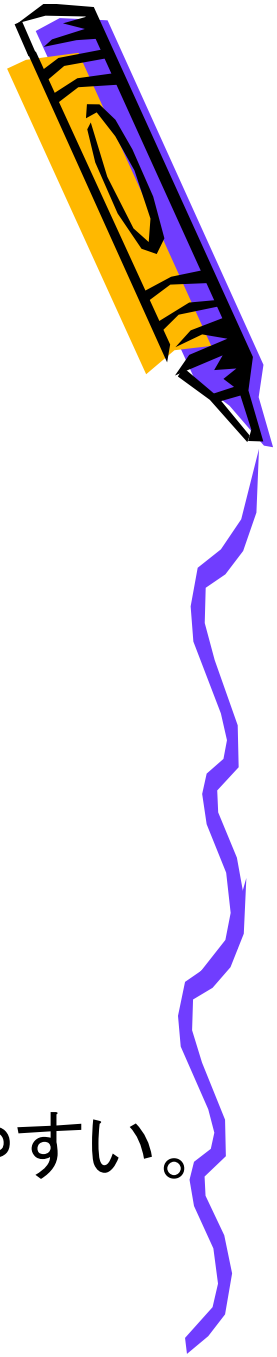


- 省消費電力に対する強い要求
  - 電池駆動時間
  - 省エネ
- 小型の機器も多い。
  - 携帯機器など
- 低価格化に対する強い要求
  - 製品の競争力のため。
  - 必要最小限のスペックを見積もり、搭載するコンピュータを選定する。
- 部品点数の削減（多機能ワンチップ化）
  - 低消費電力・小型化・低価格化

組み込みシステムに利用されるコンピュータ

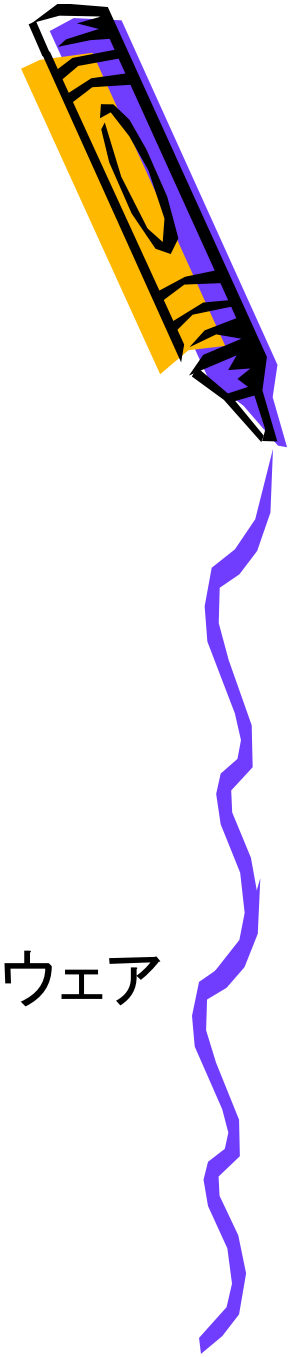
- 周波数：数MHz～数百MHz
- メモリ：128バイト程度～数十Mバイト
- 電力：数mW～数W
- 不揮発性メモリ：ROM, StaticRAM, フラッシュメモリ
- 多機能ワンチップ化の例：
  - AD/DA、RAM、フラッシュROMなどを搭載
  - シリアル、USB等の通信用インタフェースを搭載
  - DSP搭載など

組み込みシステムのソフトウェアの特徴



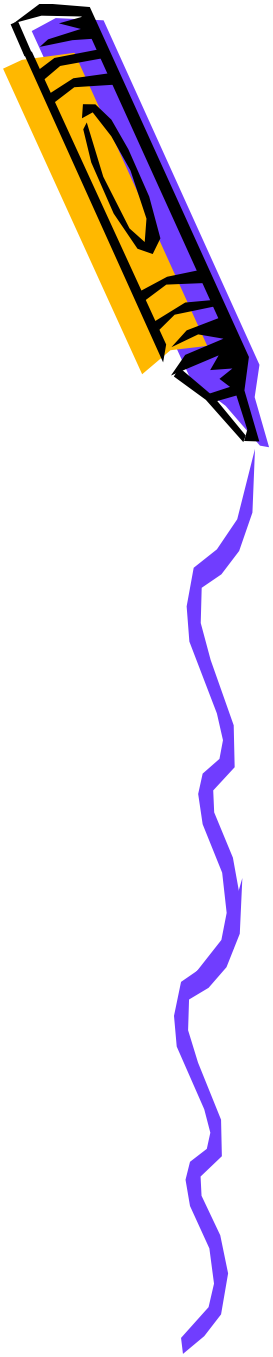
- コードサイズは比較的小さい。
- 個々の処理は比較的単純。
- 割り込み（イベント）応答速度が重要。
- 開発期間短い。
- 使用期間長い。
- バージョンアップは難しい。
- CPU能力の限界付近で動かすようにする。
- 個別の製品・対象に特化したプログラムになりやすい。

組み込みシステムの開発



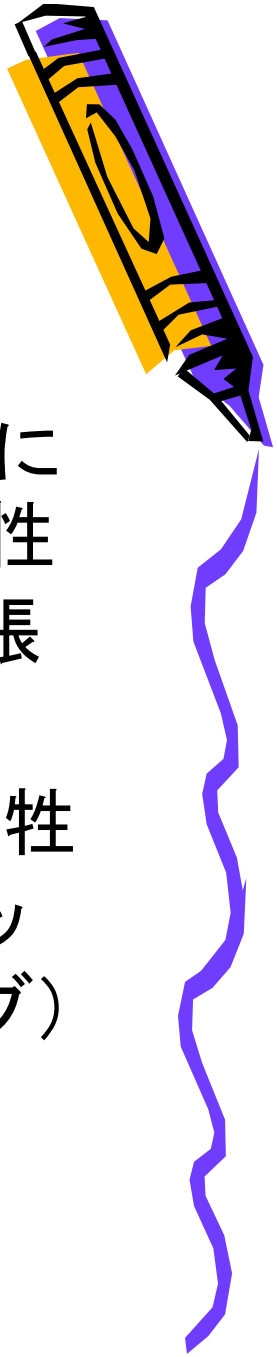
- 過去：
  - 非力なプロセッサ。わずかなメモリ。
  - アセンブラ多用
  - C言語と割り込みを利用するプログラム
  - 貧弱なデバッグ環境
  - 職人芸
- 過去→現在→未来：
  - 昔のPC並みの性能のプロセッサ
  - 昔のPC並みのメモリ容量
  - オブジェクト指向言語の利用→再利用可能なソフトウェア資産（ソフトウェア部品）
  - OSの導入
  - ICE（In Circuit Emulator）

組み込みOS



- 必要な機能を最小限に絞ったOS
- 豊富なコンフィギュレーション機能
- 最小構成～最大構成まで、多様な構成
  - カーネルサイズ
  - 必要メモリ
  - ライブラリ選択
  - デバイスドライバ選択
  - ファイルシステムのサイズ
  - etc…

組み込みOSの必要性



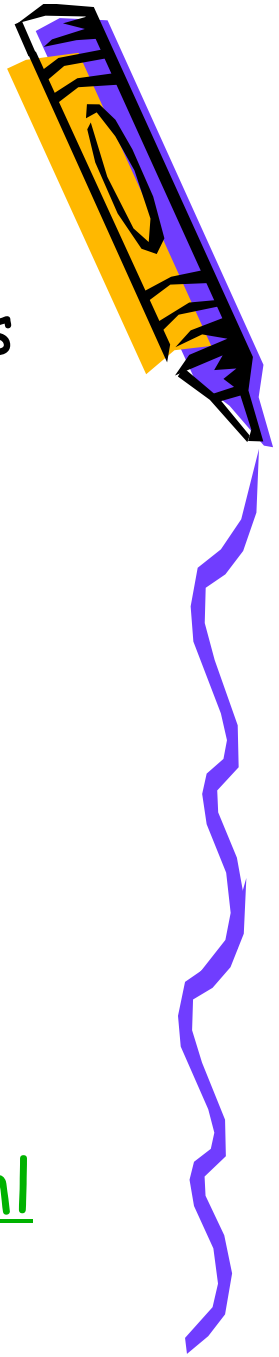
小規模のシステムにおいて、

- アセンブラや、割り込みを利用して、職人芸的に書かれてきたプログラム　→　開発効率・生産性向上、ソフトウェアの可読性向上、保守性・拡張性向上
- OSの使用リソースを最小限に抑え、性能を犠牲にしないことが求められる。（メモリにもプロセッサ能力にも余裕の無い環境でのプログラミング）

組み込みOSの例

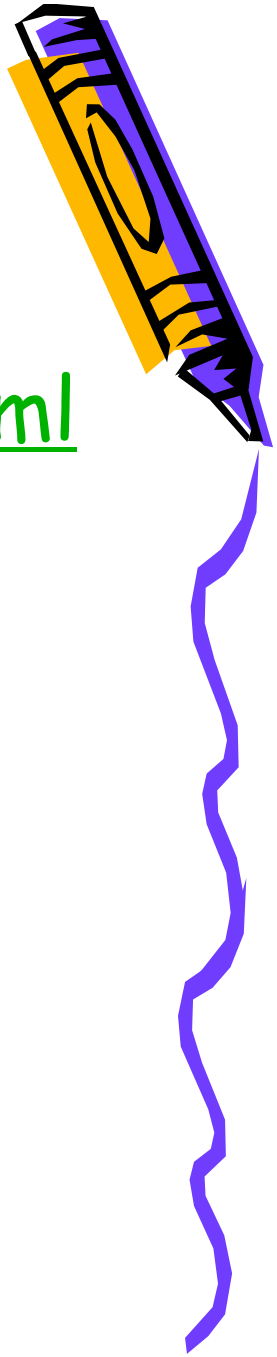
- uClinux
  - http://www.uclinux.org/
  - Linux環境に類似（Linuxのサブセット）
  - カーネルサイズ500kB～
- eCos （Embedded Configurable Operating System）
  - http://sources.redhat.com/ecos/
  - 豊富なコンフィギュレーション機能。リアルタイム拡張も。
  - カーネルサイズ60kB～
- ITRON （Industrial TRON）
  - http://www.ertl.jp/ITRON/home-j.html
  - リアルタイム機能を持つ。
  - 仕様のみ策定。実装は多様。
  - カーネルサイズ数kB

TRON Project



- http://www.tron.org/
- TRON: The Real-time Operating system Nucleus
- 1984年～ 東京大学 坂村健
- コンピュータの体系を再構築する。
- オープンアーキテクチャ
- リアルタイムOS
- 仕様のみを策定し、実装は他者
- 弱い標準化
- ドキュメントは、
  http://www.assoc.tron.org/jpn/document.html

ITRON

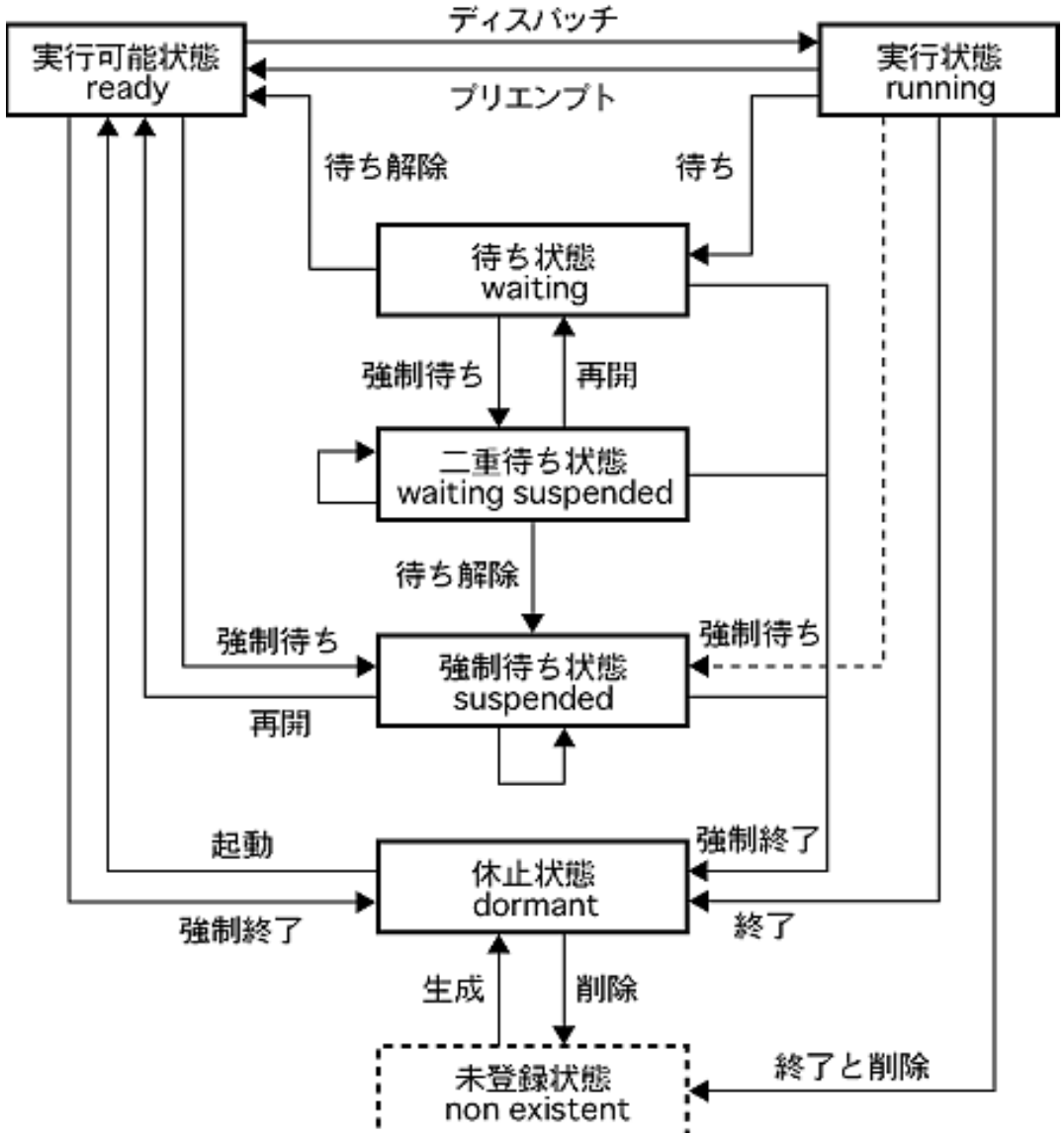


- Industrial TRON
- http://www.ertl.jp/ITRON/home-j.html
- 機器組み込み制御システム用

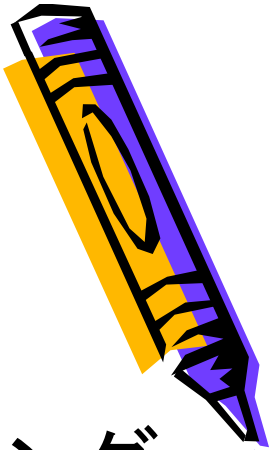
μITRON仕様におけるタスク状態

1. 実行状態（RUNNING）（RUNだった時もある）
2. 実行可能状態（READY）
3. 広義の待ち状態
    1. 待ち状態（WAITING）（WAITだった時もある）
    2. 強制待ち状態（SUSPENDED）（SUSPENDだった時もある）
    3. 二重待ち状態（WAITING-SUSPENDED）（WAIT-SUSPENDだった時もある）
4. 休止状態（DORMANT）
5. 未登録状態（NON-EXISTENT）

μITRON仕様におけるタスク状態遷移



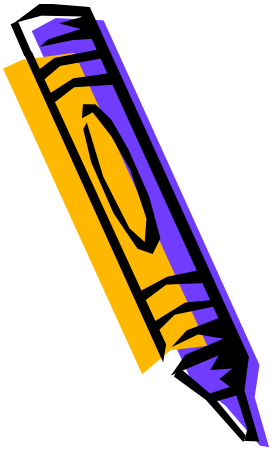
μITRON仕様における スケジューリング



- プリエンプティブな優先度ベーススケジューリング
  - プリエンプティブ：横取り可能
- 優先度の高いタスクが実行中である限りは、それより優先度の低いタスクは全く実行されない。
  - TSS（Time Sharing System）ではない。
- 同じ優先度を持つタスク間では、FCFS（First Come First Served）方式によりスケジューリングを行う。

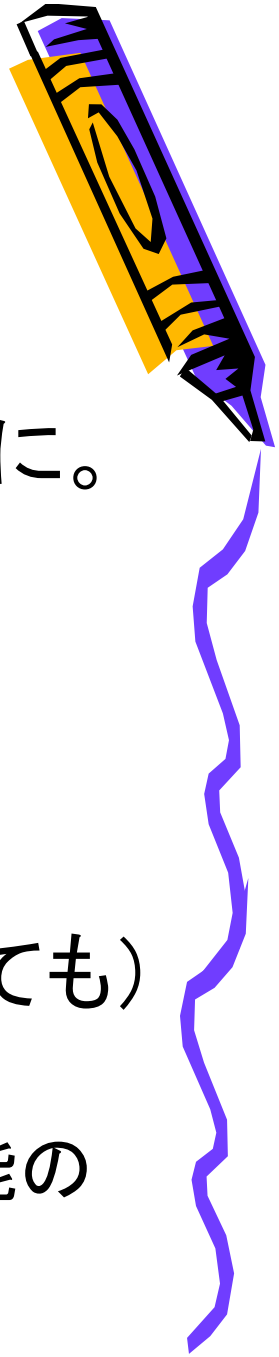
μITRON仕様カーネルの実装

TRON協会は仕様のみを策定。実装は様々。



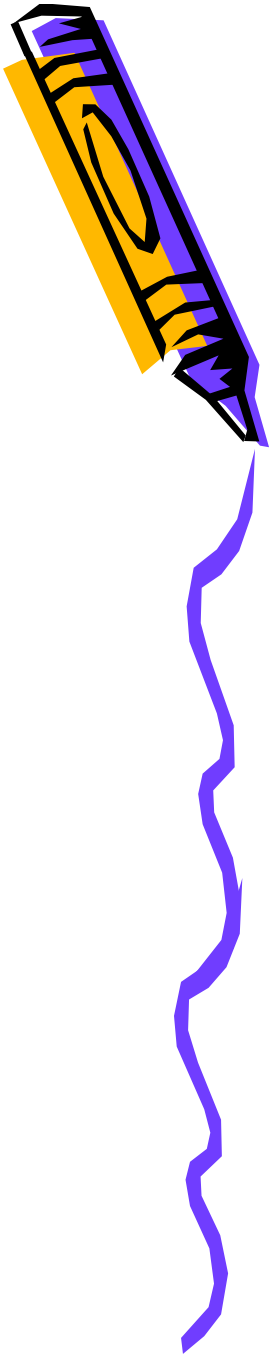
- 商用
  - ITRONのホームページには、トロン協会に登録されたITRONカーネル一覧が記載されている。
    http://www.assoc.tron.org/jpn/product/list-j.html
- フリー
  - フリーで公開されているものも多い。
  - 例：TOPPERS/JSPカーネル
    - http://www.ertl.jp/TOPPERS/
    - ターゲットプロセッサ：M68040, SH1,3,4, H8, H8S, ARM, V850, i386, ...
- 社内独自実装
  - 電気・電子機器メーカが自社製品用に実装して利用しているものは、非常に多い。

HOS (Hyper Operating System)



- 個人で作成しフリーで公開している μITRON3.0準拠RTOS
- sourceforgeに移行し、グループで開発に。 μITRON4.0準拠。
- ウェブページ
  - http://hos.sourceforge.jp/
  - https://sourceforge.jp/projects/hos/
- カーネルサイズは（全ての機能を利用しても）100kB未満
  - ライブラリの形なので、実際に利用する機能のコードのみのサイズになる。

HOS-H8



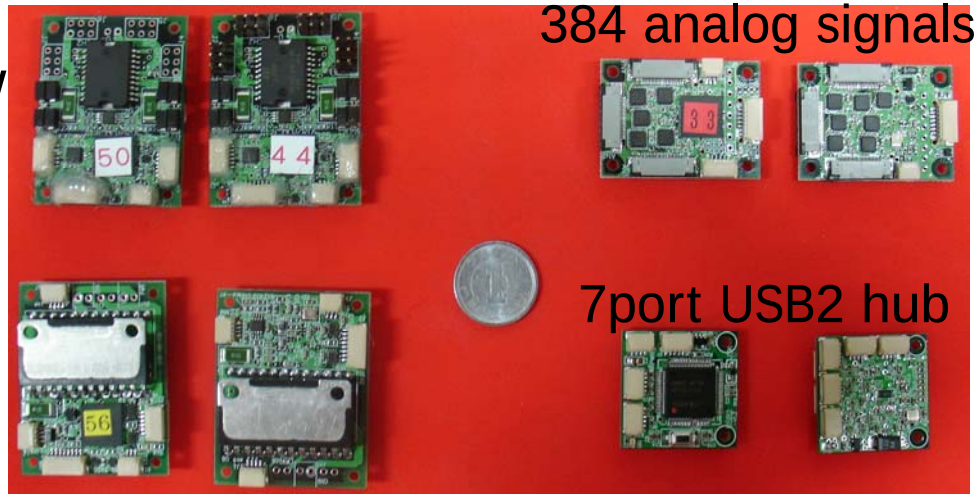
- 日立 H8/300H 用のμITRON3.0仕様のレベルSまでをサポートしたリアルタイムOS
- 秋月電子の基板に付属するCコンパイラやアセンブラに対応した形
- gccクロスコンパイラを利用する環境も第三者によって整えられている。

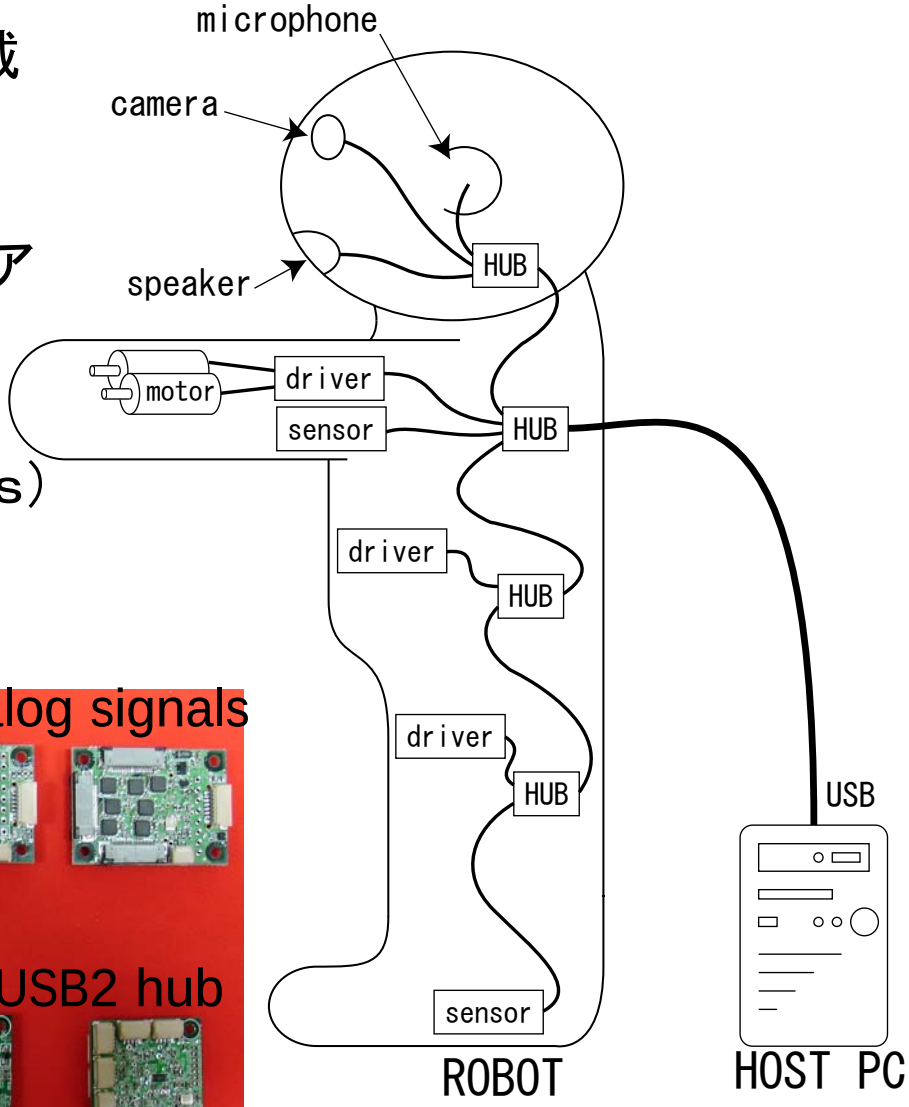
超多自由度筋骨格型ヒューマノイド：小太郎・小次郎

- 超多自由度ヒューマノイド
- 全身腱駆動
- 柔軟性を持つ脊椎
- 可動範囲を広げる肩甲骨
- 全身でモータ約100個
- 張力、長さ、電流、温度
- 分布触覚センサ
- 姿勢センサ
- 眼球（カメラ）、マイク、スピーカ
- 約50個の分散プロセッサ
- 身長125[cm], 130[cm]
- 体重25[kg], 45[kg]

小太郎の通信・制御システム

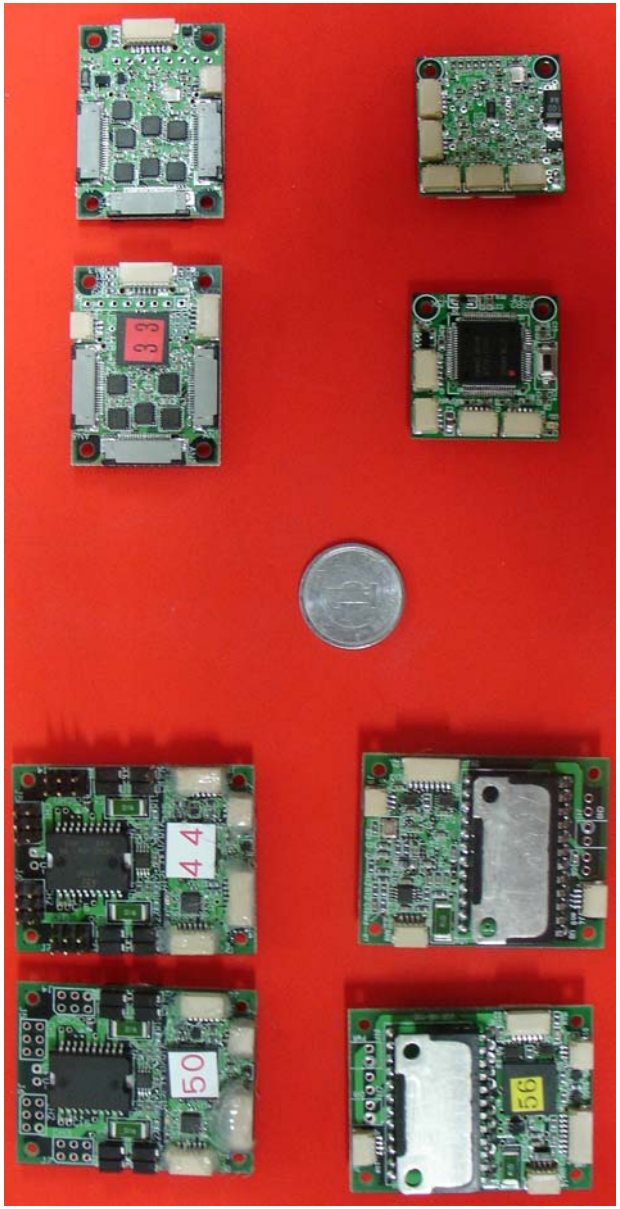
- マイコン・モータドライバ・AD変換器搭載基板
- 制御は基板上で
  - 電流制御・・・回路又はファームウェア
  - 電流制御・モータ角度制御・力制御
    ・・・ファームウェア
- 制御目標値更新はUSB（周期1～16ms）
- 上位計算機からはUSB線1本のみ
- 視聴覚用デバイスと共存





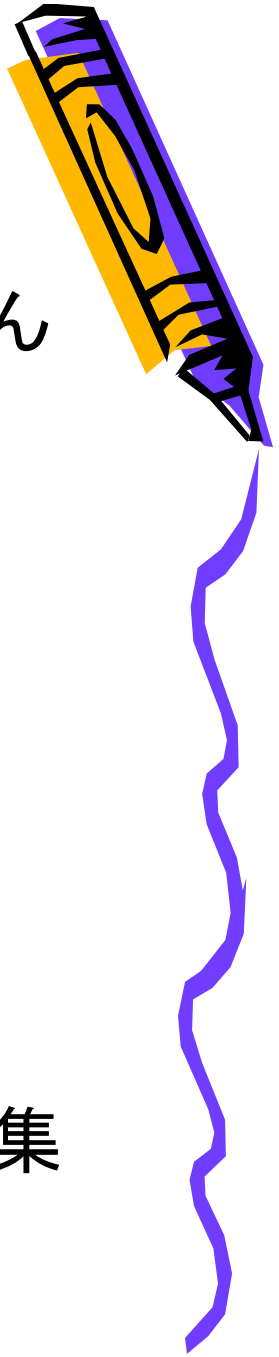
USBを利用した体内制御系を開発

- 共通仕様
  - USB1.1回路内蔵 Renesas H8S/2215
  - シリアルポート×３
  - etc
- モータドライバ基板１（４．５Wモータ用）
  - ４軸（各軸定常2A最大3Aまで）
  - 電流監視。張力・温度センサ接続
  - 36mm×46mm、USB1．１等
- モータドライバ基板２（20Wモータ用）
  - ２軸（各軸定常5A最大15Aまで）
  - 電流監視。張力・温度センサ接続
  - 36mm×46mm、USB1．１等
- 触覚センサ用基板
  - 384点のアナログ信号を計測
  - 全身用触覚センサ・姿勢センサ等用
  - 28mm×38mm、USB1．１等



ソフトウェア第三　水内郁夫

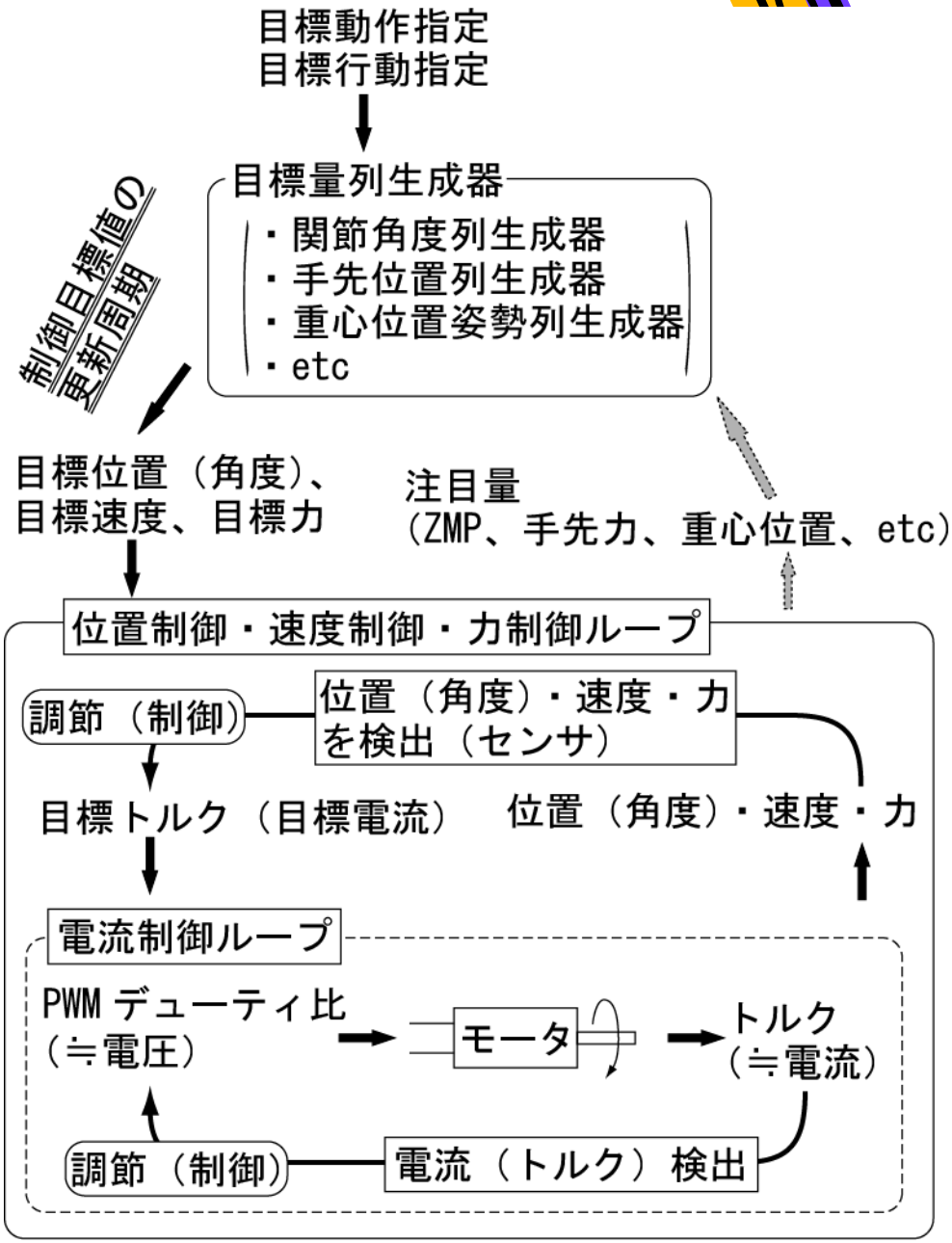
なぜ分散型か？



- 分散以外の選択肢は：
  - 集中型：例えば、一台のPCにIOボードをたくさん挿す。
- 拡張性：
  - センサやアクチュエータを追加・変更しやすい。
  - 局所的な反射機能の付与など。
- 規模：
  - 集中型ではPCのスロットが足りなくなるかも？
- 配線：
  - 集中型では、全ての配線が集中コントローラに集まる。

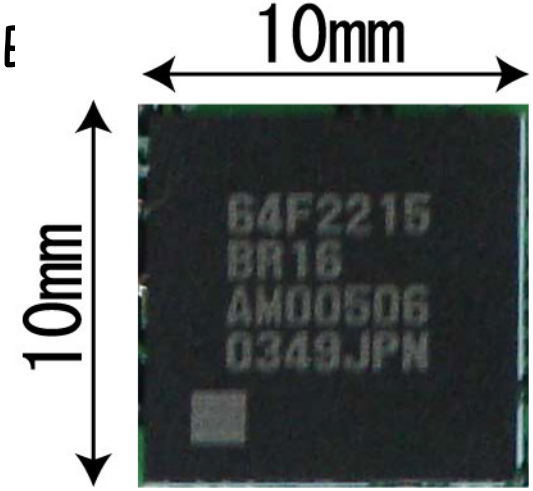
ロボットの制御システムの構成

- 何の制御をどこで行うか？
  - 電流制御
  - 位置制御
  - 力制御
- 制御目標値更新
  - 周期は？
  - 方法は？
  - 制御目標生成法は？



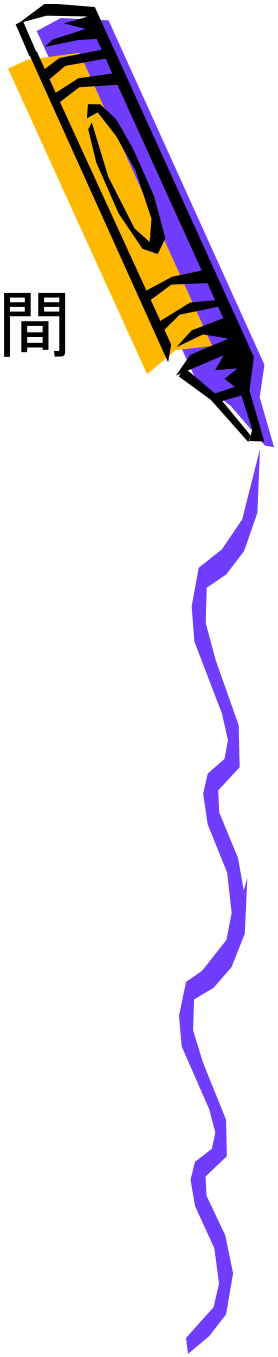
体内分散基板のプロセッサ

- H8S/2215（ルネサステクノロジ）
  - 16bit、16MHz、256kB Flash、16kB
  - BGA 112pin
  - 整数乗除算命令
  - USB1.1 (12Mbps)、シリアル 3ch
  - AD 8ch / DA 2ch
  - PWM 7ch
  - タイマ各種
- プログラム
  - μITRON互換OS
  - gccでクロス開発
  - 制御周期0.5～25.5[ms]（周期は可変）
  - 筋の長さ制御、張力制御などの、制御モードを切り替え
  - データの更新は16[ms]（PCとUSBで通信）
  - タスク：制御、AD、シリアル通信（対話的なモニタ）
  - 割り込み：USB、シリアル、など



H8S/2215（BGA）

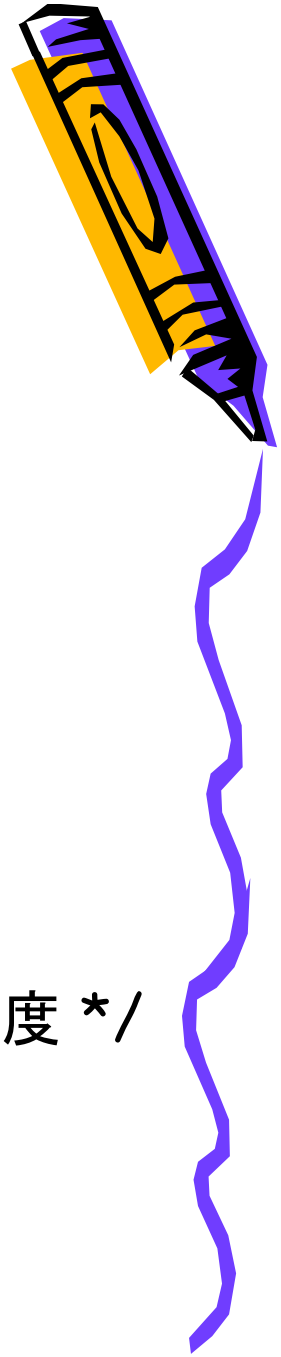
μITRONを利用するプログラミング（1）



- タスク毎に周期実行する
  - get_tim()（時間取得）と dly_tsk()（指定時間待ち）を組み合わせる。

```
void servo_task(INT arg) {
  while (1) {
    get_tim(&t1);
    motor_out(calc_servo(get_sensor()));
    get_tim(&t2);
    dly_tsk(cycle – (t2-t1));
  }
}
```

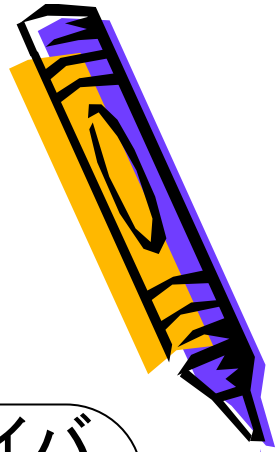
μITRONを利用するプログラミング（2）



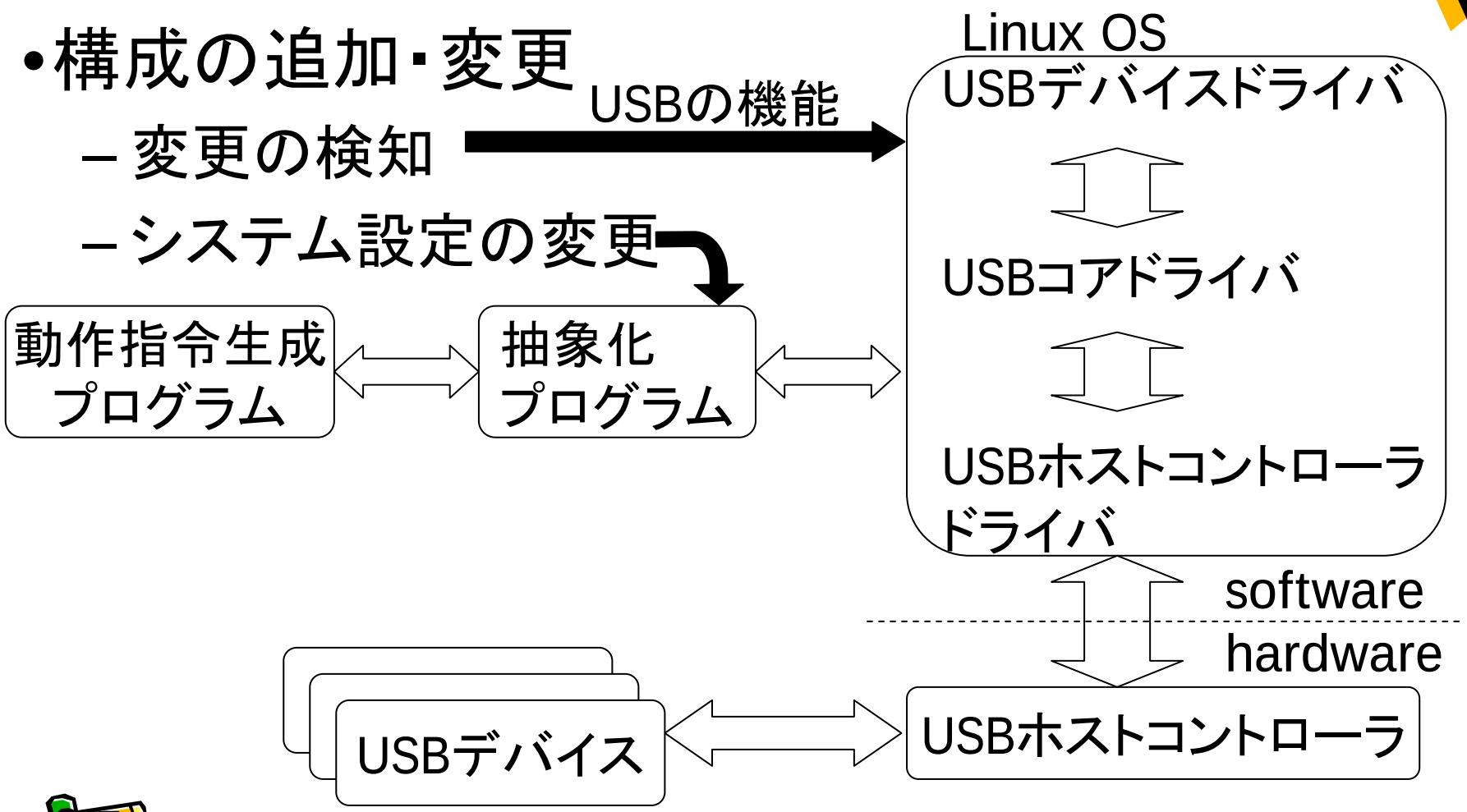
- 各タスクで、仕事の必要があるかをチェックし、無ければ他タスクに譲る。
  - rot_rdq(); を利用する。

```
void ad_task(INT arg) {
  while (1) {
    if (ad_finished()) {
      read_ad_data();
    }
    rot_rdq(TPRI_RUN);  /* TPRI_RUNは、自分の優先度 */
  }
}
```

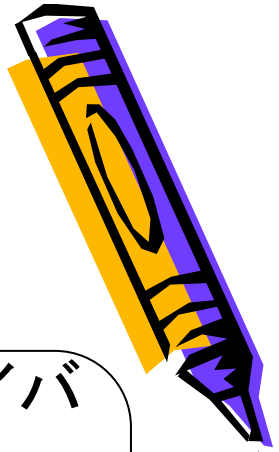
拡張性の高いロボット制御ソフトウェアシステム



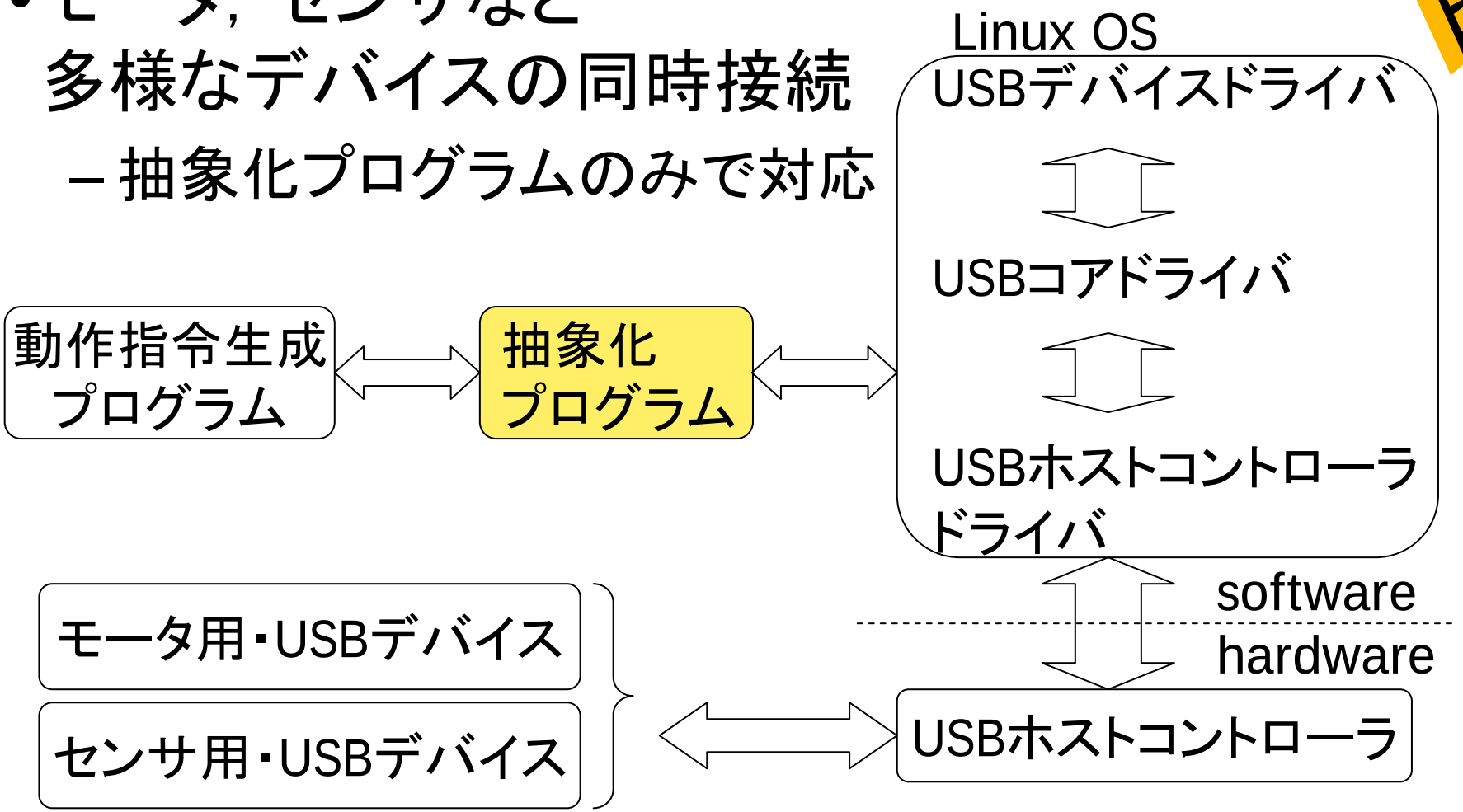
- 構成の追加・変更
  - 変更の検知
  - システム設定の変更



多様なデバイスの抽象化

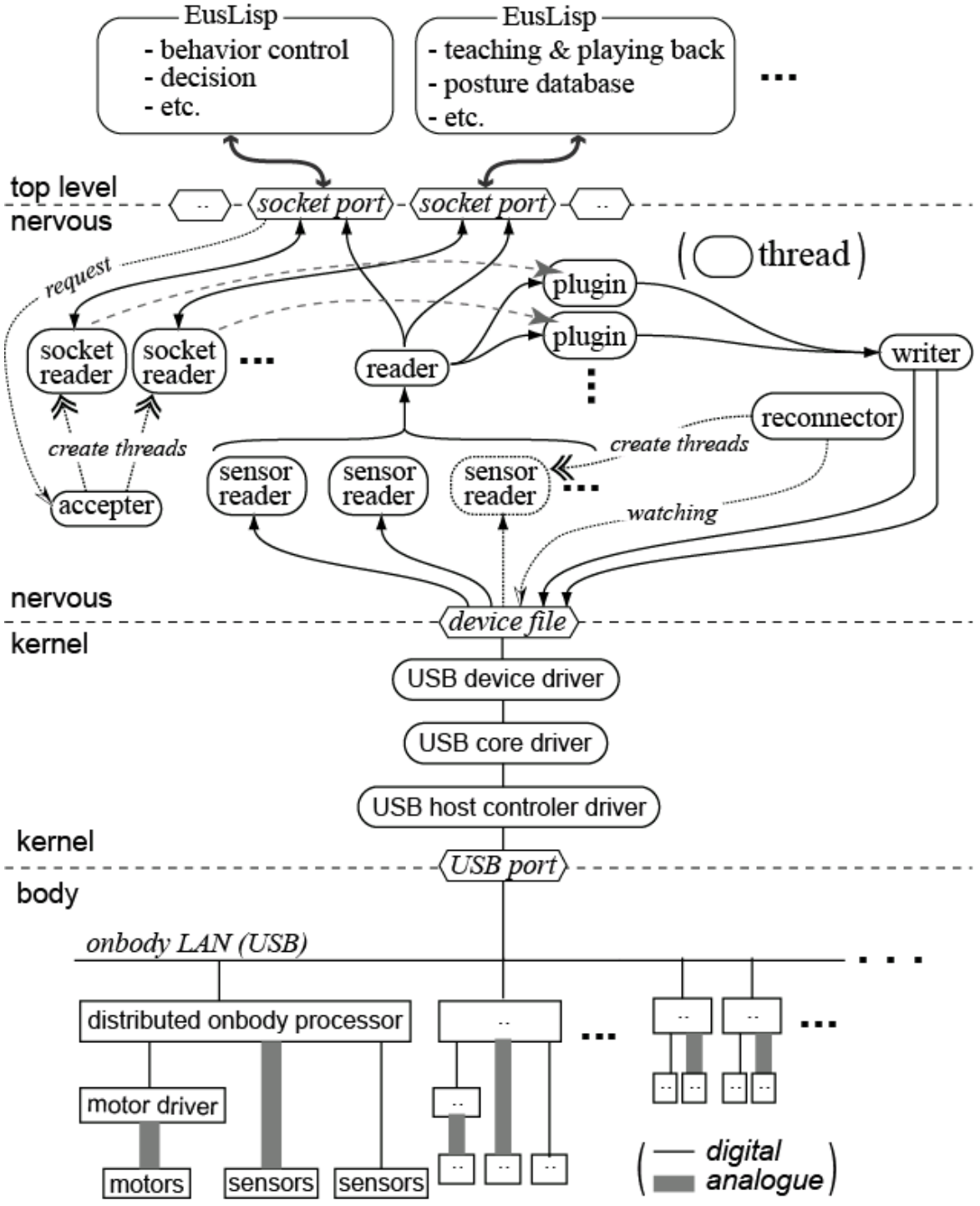


- モータ，センサなど
  多様なデバイスの同時接続
  - 抽象化プログラムのみで対応



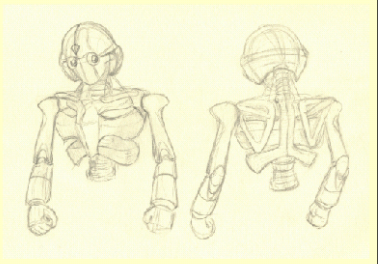
小太郎の全体システム

- 体内分散CPU基板
  - モータドライバ
  - センサ
  - USB
- 体外のPC（Linux）
  - USBのデバイスドライバ作成
  - 中間層:"nervous system"
    - 体内とUSBで通信
    - 上位層とソケット通信
    - センサ・モータ値をいじる周期的なプラグイン
  - 上位層：EusLisp
    - 状況認識
    - 行動選択・制御



超多自由度・可変柔軟脊椎・筋骨格型ヒューマノイド 小太郎

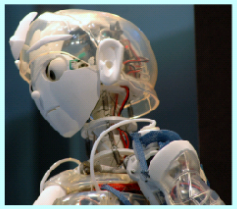
デザイン



全身をＲＰにより製作

超多自由度 筋配置可変な身体

筋駆動マルチメディアヘッド



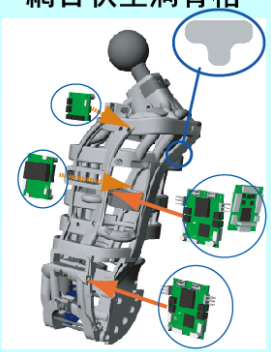
- 筋駆動眼球
- USB2カメラx2
- USBマイクx2
- USBスピーカ
- ジャイロx2
- 3軸加速度センサ

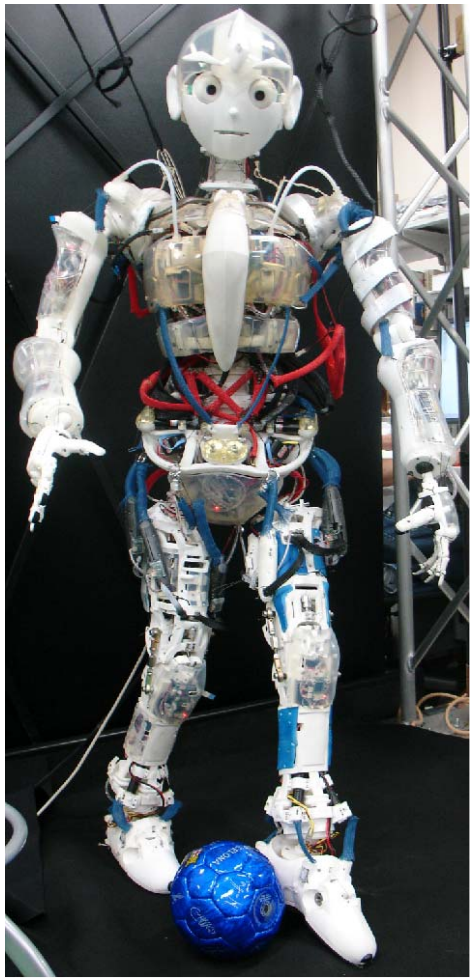
弾性要素・粘性要素

- バネ,シリコンゴム
- 初期姿勢への復元力（筋を補助）
- 安全性
- 滑らかな変形形状
- 脊椎・頚椎の各節

肩甲骨・鎖骨構造

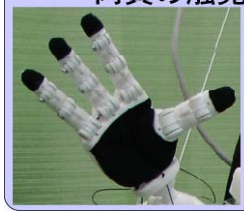
網目状空洞骨格





携帯電話用カメラを使った球関節姿勢センサ

肉質の触覚センサ

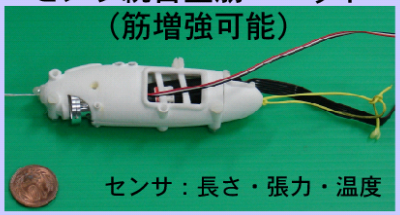


- 導電性発泡体
- 電極を3次元的に配置
- 電極間の抵抗変化計測
- 触覚ＲＰ

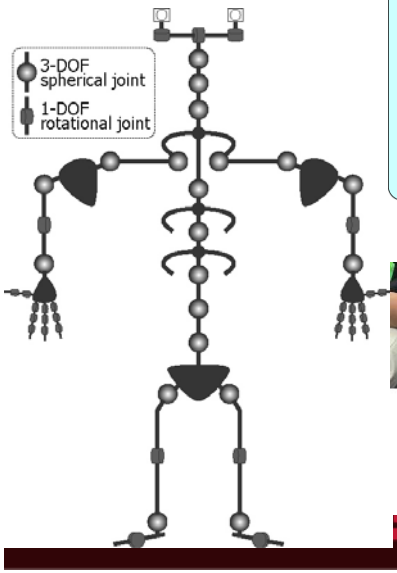
包帯状触覚コンポーネント

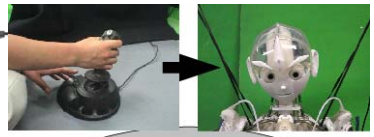
- 多様な配置可能
- 8x8のマトリクス
- フレキシブル基板＋感圧導電性ゴム
- ＬＥＤ
  - 逆流防止
  - 触覚力視認

センサ統合型筋ユニット（筋増強可能）

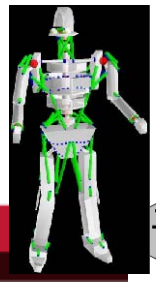


多種多数のセンサを擁する身体

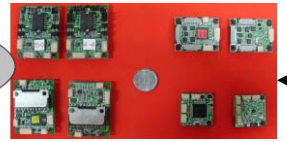




ジョイスティックによる操作システム



超多自由度身体のための感覚動作制御通信システム

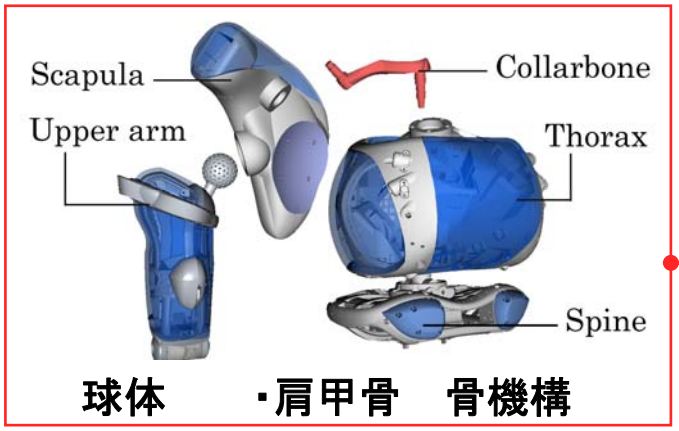


エクセルにより設定可能な筋骨格幾何モデル

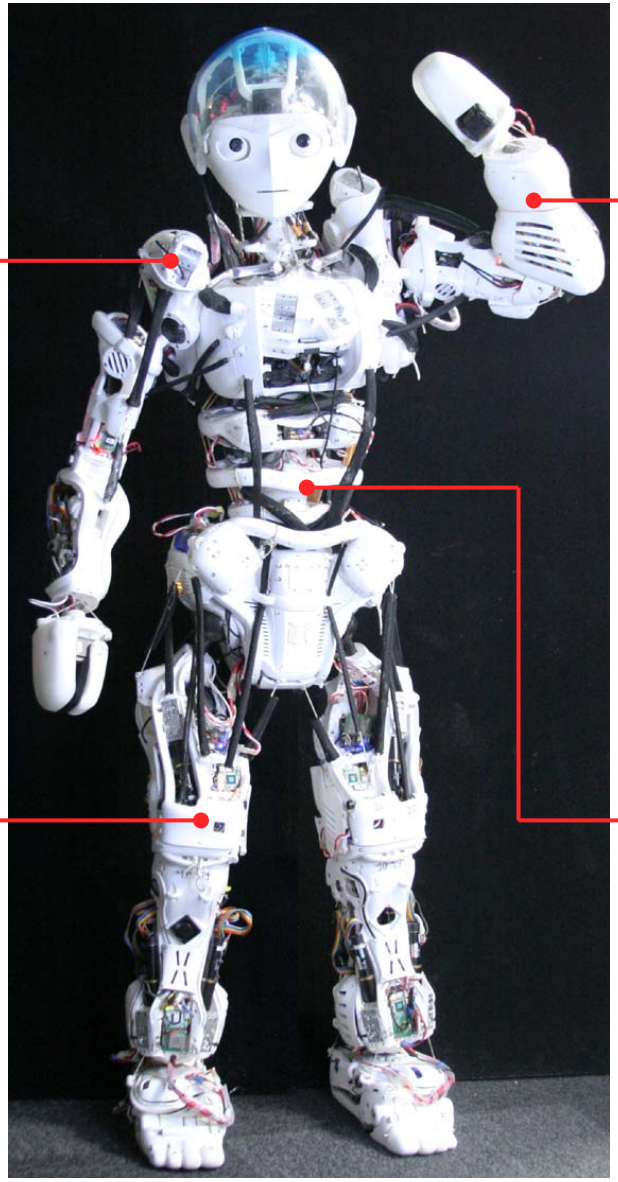
並列評価監視構造による自律行動統合システム

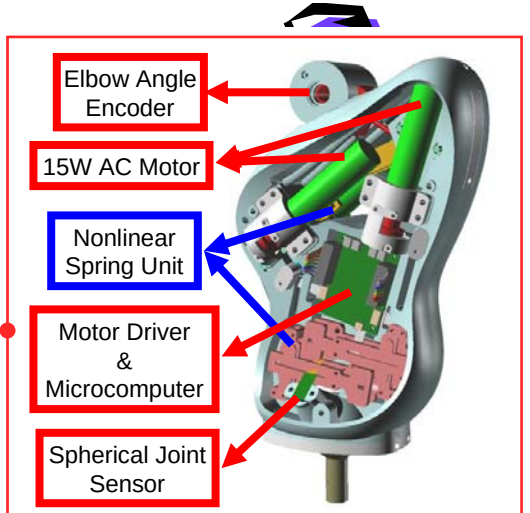
- 球　におけるデモの様子

筋骨格型ヒューマノイド　小次郎



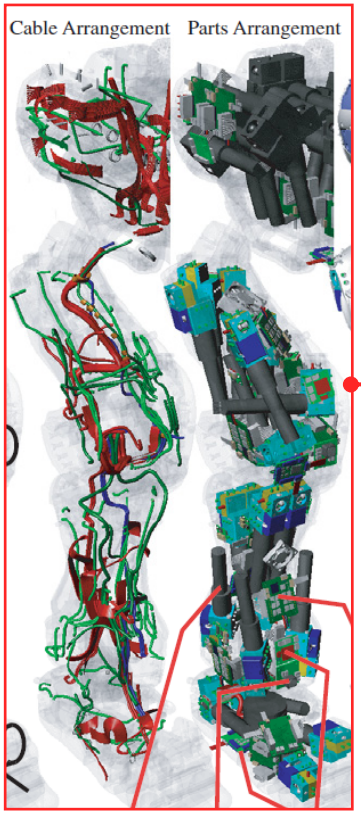
球体　　・肩甲骨　骨機構



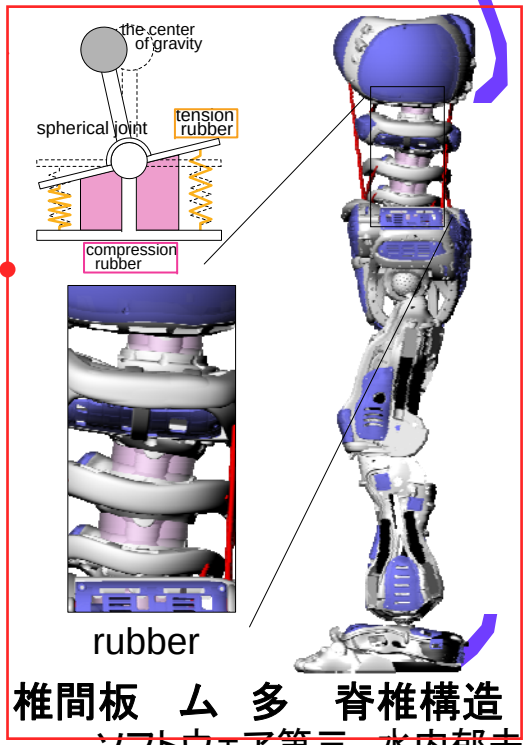


性可変非線形バネ
ユニット搭載　　構造

自由度配置



部の内部配線，部品配置



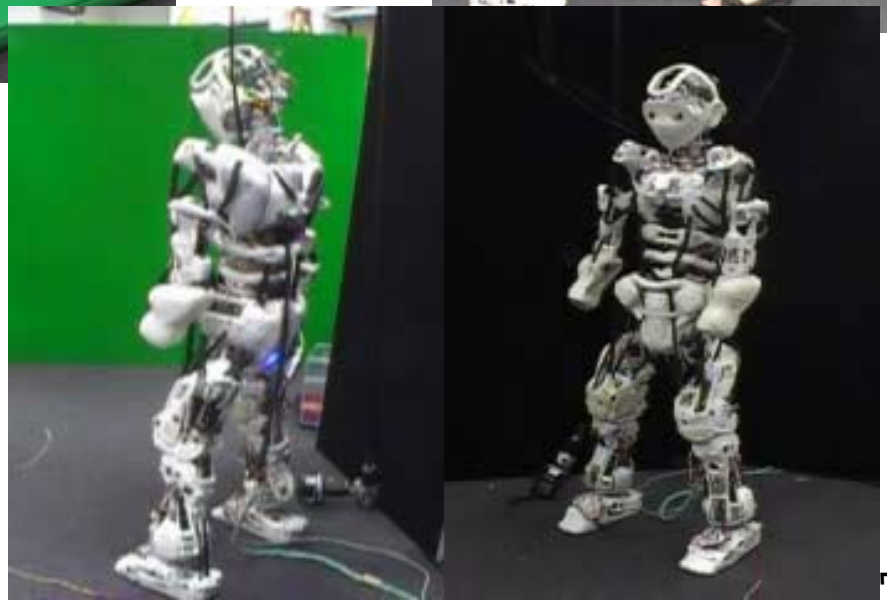
椎間板　ム　多　脊椎構造

ソフトウェア第三　水内郁夫

筋骨格型ヒューマノイド 小次郎 における全身運動例

起き上がり運動

反射運動を用いた　行



骨と肩甲骨を　用した
全身動作