Multi Thread Tiny BASIC

たけおか

take@takeoka.net

@takeoka

Tiny BASICについて

Tiny BASICの特徴

- 1977～1983年頃 大流行
- 小メモリで動作
  - 処理系(インタープリタ)： 2K Bytes程度
  - アプリケーションは、残りのRAM
- フロッピ・ディスク、ハードディスクなどの外部記憶装置が無い時代
  - カセットテープのみが外部記憶
  - ファイルシステムが無い
- 対話的に使える
  - モニタ・プログラム代わりに便利
    - shell scriptのように、プログラムを書く
    - プログラムは、使い捨て
    - 電卓にもなる
    - IO装置やメモリのテストに便利
- エディタも内蔵
  - 行単位の編集

Tniy BASICの色々

- Palo Alto Tiny BASIC
- なかもず Tiny BASIC
- 電大版Tiny BASIC

- SWTPC 4K BASIC

- GAME
  - キーワードが記号
  - 少ないキーストロークで入力
  - 小メモリ

Tniy BASICの実現方法

- プログラムは、変換せず、文字のままメモリに格納
  - 変換系が要らない
  - 編集系が楽
  - 行番号だけは二進数値化

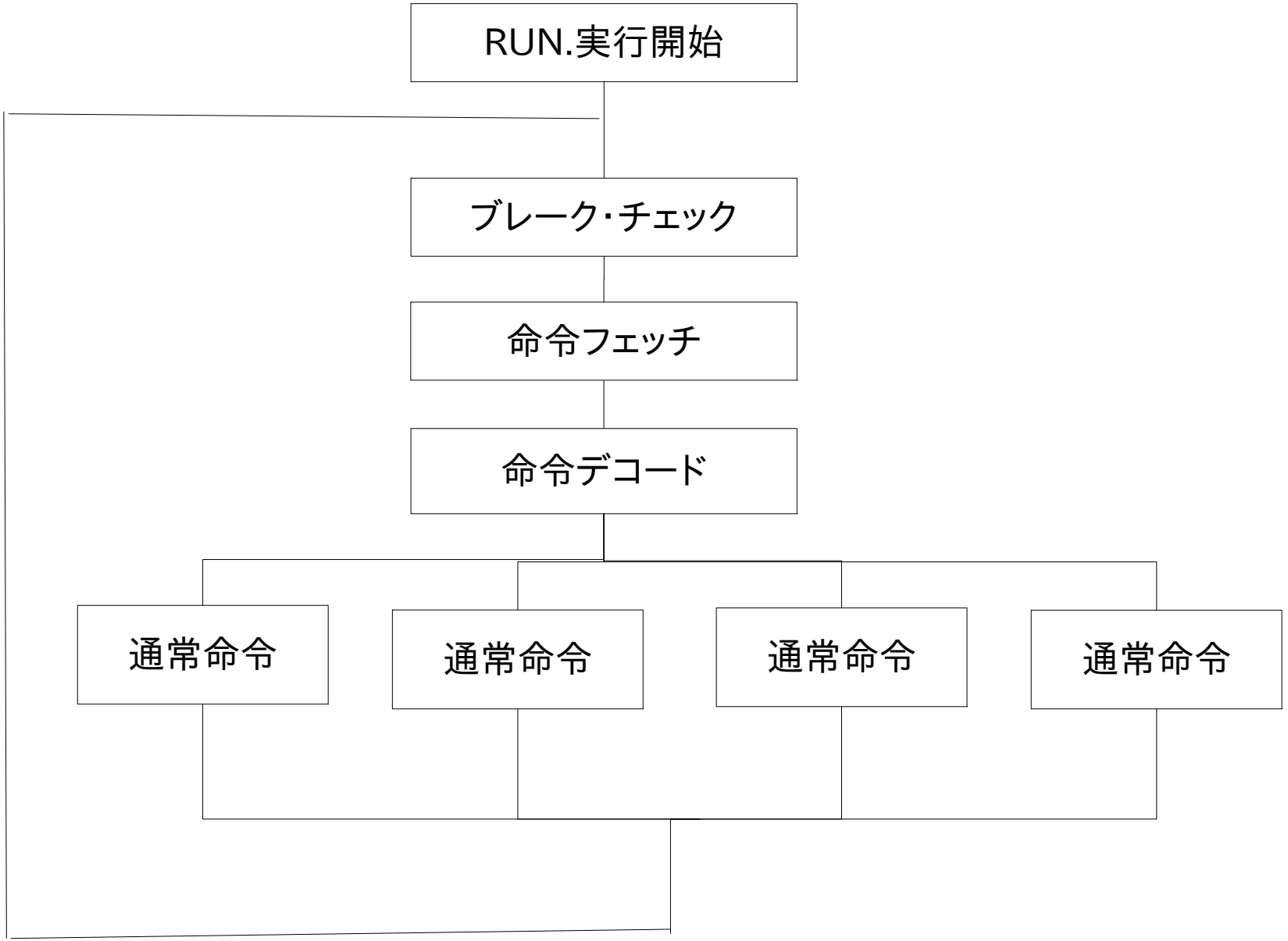
    ※Tiny ではない BASICは、プログラムを中間コードに変換して格納していた
    - Apple][6K BASIC
    - Microsoft M-BASIC (4～8KBytes 程度)
- インタープリタ
  - メモリ上の文字列を、実行時に解釈して、実行

インタープリタ

- 仮想機械
  - ソフトウェアで実現された機械
  - Tiny BASICの機械語は、文字列なのでダサい
    - 可変長の命令セットを持つ、美しくない x86 は、みんな使っている
  - 機械語命令が可変長の文字列であるハードウェアは実在した
    - IBM 1401
    - http://en.wikipedia.org/wiki/IBM_1401

インタープリタ

- 仮想機械の実行



マルチスレッドとは

マルチ・スレッド

- スレッドとは
  - 計算に必要なコンテキストを持ち、CPUが割当たると計算を実行
    - コンテキストとは
      - PC, StackPointerなどのレジスタ
      - スタック
- マルチ・スレッド
  - スレッドが並行に走行
    - 並行とは

      同時 (並列) であるかも知れないし、時分割かも知れない

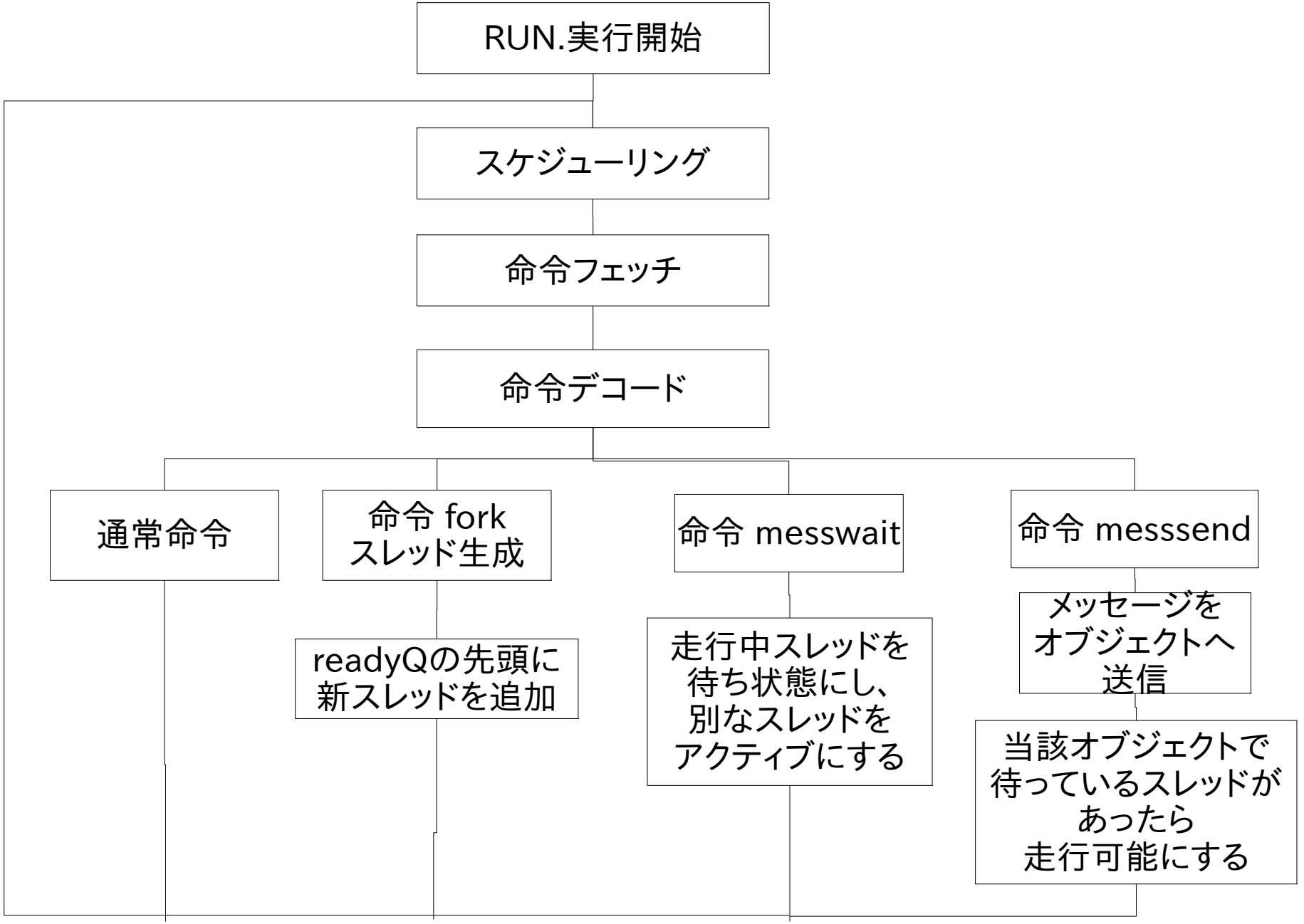
      概念的な言葉
  - メモリ空間は、スレッド間で共有

マルチ・スレッド BASIC

- スレッド・コンテキスト
  - PC, StackPointer レジスタ（レジスタは2個のみ）
  - スタック： 独立した場所に確保
- メモリ空間は、スレッド間で共有
  - 全変数をスレッド間で共有

マルチ・スレッド・インタープリタ

- マルチスレッド仮想機械



マルチ・スレッド
Tiny BASIC

マルチ･スレッド BASIC 命令語

- fork(行番号)
  - 式を開始行番号として、threadを生成する
  - スレッドのプロセス番号を返す
- end
  - メインスレッド(プロセス番号1)ならば、全スレッド停止
  - 子供スレッドならば、そのスレッドのみが終了する
- join ※未実装
  - スレッドの待ち合わせ
- sleep 式
  - 式の値で示された単位時間だけ、スレッドの実行を停止する

マルチ･スレッド BASIC 命令語

- messnew(式)　関数
  - メッセージ･オブジェクトを作る（式はダミー）
- messwait 式1,式2
  - 式1の値で示される番号のオブジェクトのメッセージを待つ
  - すでに、そのメッセージを待っているプロセスがあった場合は、待たずに抜ける
  - 式2でタイムアウト時間を指定
  - 0番のオブジェクトは、キーボード入力
- messread(式1)　関数
  - 式1の値で指定されるメッセージ･オブジェクトから読み出し
- messsend(式1,式2)　関数
  - 式1の値で指定されるメッセージ･オブジェクトへ、式2の値をメッセージとして送る
  - そのメッセージ･オブジェクトを待っているプロセスがあれば、プロセスは走行可能になる

マルチ･スレッド BASIC 命令語

- lastwait(式)　関数
  - 式の値で指定されるメッセージ･オブジェクトの待ちから抜けた要因を返す
    - 1: O_RES_RECV メッセージを受信した
    - -2: O_RES_TO　タイムアウト
    - -3: O_RES_WAITING 現在も待ちプロセスがある(待ちは外れていない)
    - -1: Error　オブジェクトは未使用

マルチ･スレッド BASIC 命令語

- `ps`
  - スレッドのステータスを表示

- `kill` 式 ※未実装
  - 式の値で示されるスレッドを、終了させる

マルチ・スレッド BASIC 実現について

- 割り込みを一切使用しない
  - キーボード割り込みを使用しない
  - タイマ割り込みを使用しない
    - 内部時計
      - ハードウェアが無い場合： ソフトウェアのループ回数で、作る
        - 当然、ゆらぎがある
      - RTC ハードウェアがある場合： スケジューリング前に、RTCを読み出す
        - 正確
  - 時間による、スレッド実行の打ち切りは無い
    - しかし、一命令ごとに、スケジューリングされる
- どうして、割り込みを使用しないか
  - おもしろい
    - 割り込みで仕事をやることなんか飽きた
  - 究極の移植性を追求
    - どんなハードウェアでも、ただのアプリケーション・ソフトウェアと同様な移植性
  - 完全に決定的
    - 外乱 (割り込み) による非決定性によるデバッグの困難さを排除 (簡単ということか？)

マルチ・スレッド BASIC 実現について

- スレッドのスケジューリング
- 各命令の実行の直前でスケジュール
  - ready (走行可能) なスレッドがあれば、切り替える
  - ポリシーは、適当ラウンドロビン
    - 現状は、少しぶれがある
- messwaitで待ちに入るとready queueから外れる
- messsendが発行され、messwaitから出ると
  - 走行可能になって、ready queueへつながる

マルチ･スレッド BASIC 実現について

- 重要な構造体
  - プロセス構造体

```
struct process {
        int pid;
        int stat;
        struct process *pre; /* activeQueue, waitQueue */
        struct process *next; /* activeQueue, waitQueue */
        struct process *toPre; /* time out queue */
        struct process *toNext; /* time out queue */
        int to; /* time of timeout */
        u_char *pc;
        int *stack;
        int  sp;
        int lno;
};
#define PROC struct process
```

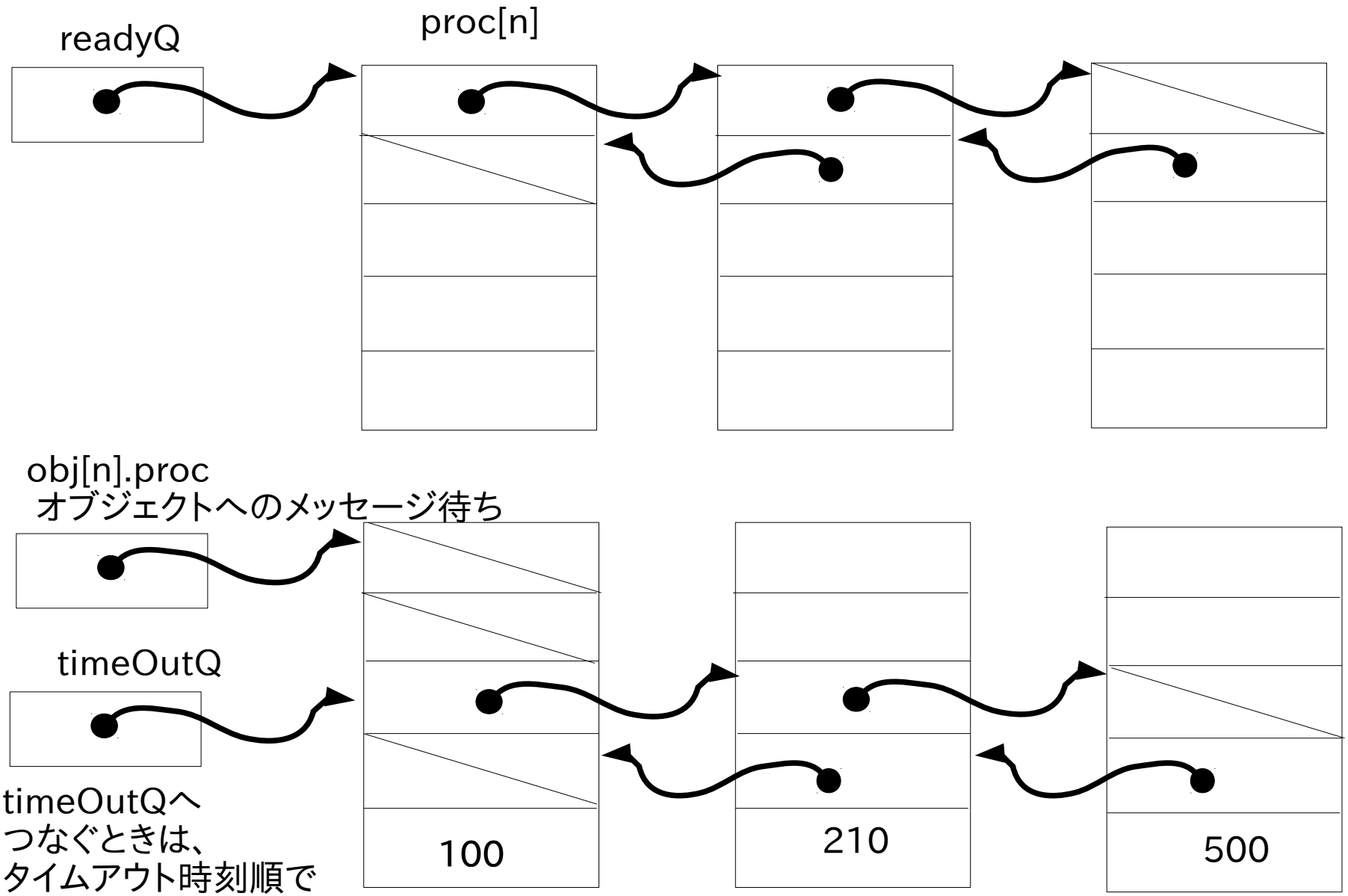
マルチ･スレッド BASIC 実現について

- 重要な構造体
  - オブジェクト構造体

```
struct object{
   int id;
   int stat;
   PROC *proc; /* waiting process */
   int value;
};
#define OBJ struct object
```

マルチ・スレッド BASIC 実現について

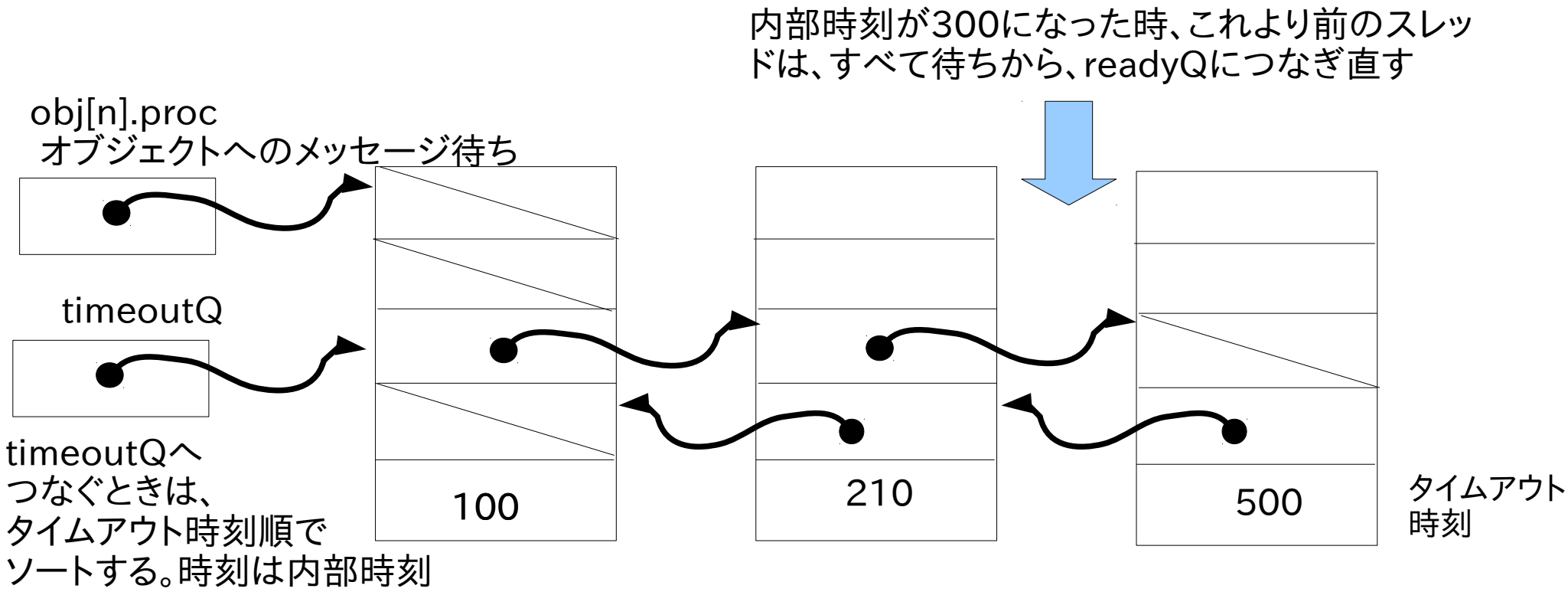
内部の重要な 待ち行列 (キュー)



マルチ・スレッド BASIC 実現について

- 時刻
  - 時刻は内部でのカウントが一定を越えたら`1tick`(単位時間)
    内部時計を進める
- タイムアウト
  - `sleep`と、`messwait`は、`timeoutQ`につなぐ
  - 内部時計を進めるときに、タイムアウトのチェックを行う

マルチ・スレッド BASIC 実現について



- オブジェクトを待ちつつ、タイムアウトも待つ
- 時刻は内部でのカウントが一定を越えたら`1tick`(単位時間)
- タイムアウトは、内部時刻(時計)を`1tick`増加させるときに、チェック
- `timeoutQ`は、タイムアウト時刻順に並んでいるので、現在時刻より以前の時刻で待っているスレッドは、すべて`activeQ`へつなぎ直す

プログラム例

```
100 ?"start" :crlf
105 j=0
107 m=messnew(0): ? "messnew=":?m:crlf
110 ?fork(300):?fork(400):?fork(500):?fork(600)
120 ?"a":crlf
130 fori=1to500000:n.
150 ? messsend(m,j) : j=j+1
170 goto 120
180 end
300 ?" messwait ":crlf
310 messwait m,100
311 ?" Wakeup=":? lastwait(m):?" "
320 if lastwait(m)<0 ?"timeout ":crlf:goto 390
330    x=messread(m): ?$ x
390 goto 300
400 ?"kkk":crlf
420 messwait 0,-1
430 ?"K "
450 y=messread(0)
460 ?$ y
480 goto 400
500 sleep 100:?"asd"
510    ps
520 goto500
600 sleep 200:?"qwe"
610    ps
620 goto600
```

実行結果

```
M>>r.
start
messnew=1
2 messwait 3kkk
45
a
asd Wakeup=-2pid, stat, PC, LNo, SP pre, next, toPre, toNext,to
0 2 0804C0DE 0 -1 080521CC 080521F8 00000000 00000000 0
1 2 0804F4FF 130 3 00000000 080521A0 00000000 00000000 0
2 2 0804F57C 311 -1 080521A0 00000000 00000000 0805227C 100
3 3 0804F5FB 420 -1 08051D40 080521A0 00000000 00000000 -1
4 1 0804F64B 510 -1 00000000 080521CC 00000000 080521F8 100
5 4 0804F666 600 -1 00000000 08052224 00000000 00000000 200
readyQ=1 0 2  current_time=100  idol_count=499996
toQ=  5:200
 obj=0 proc=00000003 stat=2  obj=1 proc=FFFFFFFF stat=1
 timeout
 messwait
0 Wakeup=1 a
00 messwait
asdqwe Wakeup=-2pid, stat, PC, LNo, SP pre, next, toPre, toNext,to
0 2 0804C0DE 0 -1 0805227C 080521CC 00000000 00000000 0
1 2 0804F4FF 130 3 080521A0 080521F8 00000000 00000000 0
2 2 0804F57C 311 -1 080521CC 00000000 00000000 00000000 200
3 3 0804F5FB 420 -1 08051D40 080521A0 00000000 00000000 -1
4 1 0804F64B 510 -1 00000000 0805227C 00000000 080521F8 200
5 2 0804F673 610 -1 00000000 080521A0 00000000 08052250 200
readyQ=5 0 1 2  current_time=200  idol_count=999978
toQ=
 obj=0 proc=00000003 stat=2  obj=1 proc=FFFFFFFF stat=1
pid, stat, PC, LNo, SP pre, next, toPre, toNext,to
0 2 0804C0DE 0 -1 00000000 080521CC 00000000 00000000 0
1 2 0804F4FF 130 3 080521A0 080521F8 00000000 00000000 0
2 2 0804F57C 311 -1 080521CC 08052250 00000000 00000000 200
3 3 0804F5FB 420 -1 08051D40 080521A0 00000000 00000000 -1
4 2 0804F64D 510 -1 080521F8 00000000 00000000 080521F8 200
5 1 0804F673 610 -1 00000000 080521A0 00000000 08052250 200
readyQ=0 1 2 4  current_time=200  idol_count=999978
toQ=
 obj=0 proc=00000003 stat=2  obj=1 proc=FFFFFFFF stat=1
 timeout
 messwait
0 Wakeup=1 a
01 messwait

M>>
```

おしまい