# イベント駆動によるリアルタイムAIエンジン

7P-3

鈴木達郎　菅原昌平
ＮＴＴ　ヒューマンインタフェース研究所

＜はじめに＞
実用的なＡＩシステムは多かれ少なかれ現実世界とのインタラクションを持っている。この現実世界とのかかわりで重要になるのがリアルタイム性である。ＡＩにかぎらずリアルタイム処理で必要なことは、緊急的な処理をいかに速く実行開始するかということである。そのためにきめ細かい優先度制御（プリエンプション）方式と、処理切換え時間の速さがシステムのリアルタイム性指標となる。[3]
さらにＡＩ応用では前もってタイミングが予測しきれない突発的なできごとに対する処理が多く[4]、非同期／非決定処理が重要である。また同じ「できごと」に対する処理でも状況によって内容が動的に変化する。

以上のような特徴を考慮した上で、従来から我々が開発しているＬＩＳＰマシン新ＥＬＩＳシステム[1]のカーネル機能に、リアルタイムＡＩエンジンとして、イベント駆動によるリアルタイムＯＳ(EXOS)を組み込んだので報告する。

1. イベント駆動

イベントとはプロセス独立に存在するobjectであり、重要な入力・状態(処理結果)に対応したものとして定義される。重要な入力や状態が生じた時（イベントが発火した時）に行うべき処理を各プロセスは前もって登録しておくことができる。これをイベントとプロセスの結合と呼ぶ。

例えば図１ではプロセスP1はイベントe3と結合しており、e3が発火した時に登録済みの関数式 (fn3 args3) を実行することになる。これをイベントe3がプロセスP1を駆動すると言う。 具体的にはプロセスP1に割込みを生じて (apply fn3 args3) がプロセスP1の環境（スタック）上で実行されることになる。また、被駆動関数 fn3 の中でイベントの持つトークン値をとり出す関数や、自分を駆動したイベントの識別子を得る関数も用意されている。なお図１に示すように、同一のイベントが複数のプロセスと（異なる関数を用いて）結合することも可能である。このとき、同時に複数のプロセスを駆動（トークンをブロードキャスト）するか、一つづつ駆動するかを指定可能である。

2. イベントの発火

イベントの発火は重要な入力や状態を検出したプロセス／処理が、イベントと結合されているプロセスとは独立に行う。
イベントの存在（識別子）さえ知っていれば、任意のプロセス、入出力割込み処理から（イベントと結合されていなくても）発火させることができる。また、発火とともにトークンとして値をひきわたすこともできる。

3. 多重イベント駆動

イベント駆動による関数実行（被駆動処理）を１つの単位と考えると、プロセス内部で複数の非同期実行単位が並行して存在することとなる。これらの実行制御

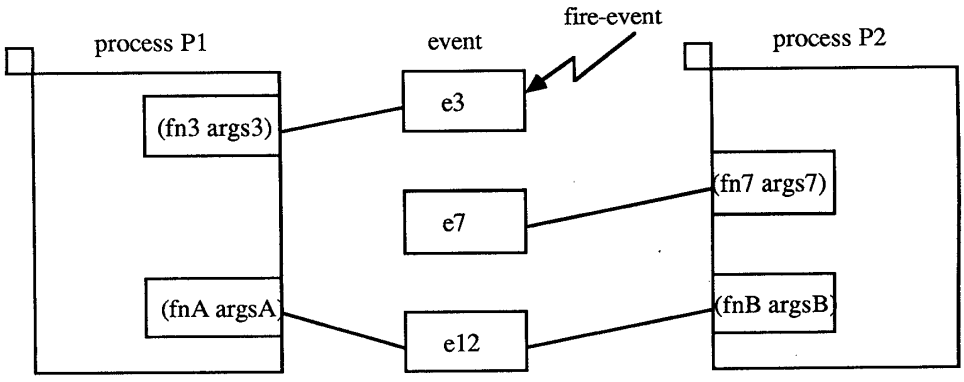


図１.プロセスとイベントの結合

Event-Driven Real-Time AI Engine
Tatsuo SUZUKI　Shohei SUGAWARA
NTT Human Interface Laboratories

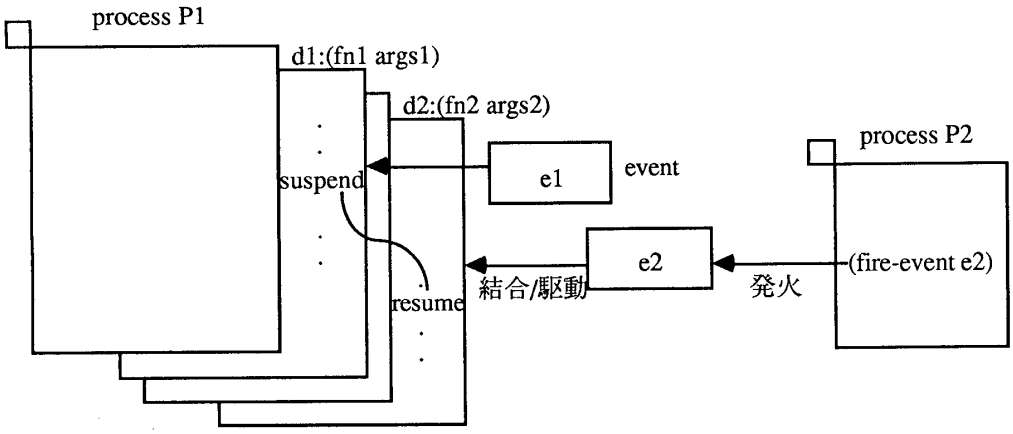


図２ 被駆動処理(d1,d2)間の同期

は通常の割込制御のように優先度により行う。即ち、新たに発生した被駆動処理が実行中の被駆動処理よりも優先度が高ければただちに実行されるが、同じか低ければ待たされる。（ただし優先度が強制モードの場合は既に強制モードの処理を実行中でも直ちに実行される。）実行権をうばわれた低優先度の処理は、上位の処理終了後に再開される。

4. 被駆動処理間の同期

ある被駆動処理 d1が別の被駆動処理 d2と同期を取りたい場合のために、同期用関数として suspend と resume がある。例えば図２に示すように d1の中で中断(suspend)しておき、d2の中でそれを再開(resume)することで同期が実現できる。

resume 時にはどの被駆動処理を再開させるかの指定と共に値 (resume value)を指定すれば、その値が suspendのリタン値となる。これを用いて非同期入出力の結果取出しなどが実現できる。suspendとresumeは同一プロセス内の被駆動処理同士の同期であるが,イベント駆動を介して（上の例ではe2を発火させることで）プロセス間の同期も可能である。なお同一プロセス内の複数の被駆動処理は独立にsuspendの状態になることができる。

5. 焦点制御

リアルタイムＡＩ処理では、状況に応じて注意を向ける範囲を限定（焦点制御：Forcusing）したい場合がある。これはイベントの有効・無効を制御することで可能となる。イベントはその優先度がいくら高くても無効にすることでそのイベントからの駆動は生じなくなる。有効にもどすまでの間、発火したトークンをためておくか、棄てるかもイベント制御機能により可能である。

また、イベント処理関数との結合や、イベント優先度は動的なものなので変更は自由である。

6. リアルタイム性能

優先度はイベントごとに定義され３２段階および強制モードからなり、各イベント駆動処理はイベントの優先度を持って走行する。

μプログラムなどの割込禁止区間をできるだけ短くすることで処理切換えに伴う割込待ち時間を短縮すると共に、プロセスに対する駆動割込は、プロセスの環境（スタックなど）そのままの上で実行させることでオーバーヘッドを少なくしている。

＜まとめ＞

リアルタイムＡＩのための基本制御機能として、イベント駆動を中心としたＯＳカーネルを作成した。

イベント駆動はプロセススイッチを伴わないよう、プロセスに対する割込処理として実現したので、リアルタイム性能がすぐれていると同時に、イベントを用いることで並行独立的な処理を記述する方法としても適している。

これらのカーネル機能はＬＩＳＰマシンＥＬＩＳの上にＬＩＳＰ関数として実現されている。今後はマルチプロセッサＯＳカーネル化を進めるとともに具体的なリアルタイムＡＩ処理に適用して評価改善を進めたい。

＜謝辞＞

本システムはＮＴＴソフトウェア研究所竹内主幹研究員の開発したＥＬＩＳ／ＴＡＯシステム[2]をベースとして用いており、ここに感謝の意を示します。また、日頃御指導をいただいている日比野主幹研究員に感謝いたします。

[参考文献]

[1] 鈴木、家吉、菅原、杉村：新ELISシステム概念，信学会、秋季全国大会、1989.

[2] Takeuchi, I. et al.: A List Processing Language TAO with Multiple Programming Paradigms, New Generation Computing, 4, 1986.

[3] 川田、安部、勝川、鈴木：分散処理用リアルタイムＯＳ，研究実用化報告, vol.36, no.9, pp1169-1175, 1987.

[4] Laffey, T. J. et al. : Real-Time Knowledge-Based Systems, AI magazine, vol.9, no.1, pp27-45, SPRING 1988.