プロセスの永続性を提供するＵｎｉｘカーネルの実装

5Ｆ－1

沖野直人*　中村嘉志†　多田好克‡
電気通信大学 大学院 情報システム学研究科§

1　はじめに

プロセスを実行途中で中断し、OSを再起動しても中断時点からプロセスの実行を継続できるような、プロセスを永続的に扱える機能は有効である。例えば、科学技術計算といった実行時間の長いプロセスでは、実行途中でコンピュータの停止が必要となる事態になっても、最初から再実行させなくてよいといった利点が生じる。

プロセスの永続性を提供する機能は有効であるが、一般的に使われているOSはこの機能を持っていない。そこで本研究ではこの機能をUnix系のOSに実装することを目的とする。

2　カーネル拡張

機能を実現するにあたっては、カーネルには手を加えないユーザレベルで実装する方法とカーネルを拡張するカーネルレベルで実装する方法がある。ユーザレベルで実装する方法の特徴を表1に示す。

表1：ユーザレベルで実装する方法の特徴

長所	○	移植性が高い
短所	×	カーネル内のプロセスに関するコンテクストを取得・復元できない
	×	既存のプログラムで機能を利用する場合、再コンパイルやリンクが必要

本研究では、以下の2つの理由からカーネルレベルで実装する。

(1) プロセスを可能な限り任意の時点で中断し、継続して実行させるためには、カーネル内のプロセスに関するコンテクストを取得・復元できなければならない。カーネル内のプロセスに関するコンテクストの取得・復元は、ユーザレベルでの実装ではできないがカーネルレベルの実装では可能である。

(2) ユーザレベルで実装した場合、一般的にプログラムの再コンパイルやリンクが必要なため、スタティックリンクのプログラムやソースコードのないプログラムでは機能を利用できない。しかし、カーネルレベルの実装では、プログラムに何の変更もせずに機能を利用できる。

3　中断・継続方法

3.1　中断方法

中断の指示には新たなシグナル（SIGXX）を追加して利用する。シグナルは非同期に発生したイベントをプロセスに伝えるものであり、中断を指示するのに適している。

中断させたいプロセスにSIGXXを送ると、シグナルを受け取った時点でそのプロセスは中断する（図1）。次にカーネルは、そのプロセスがシグナルを受け取った時点から実行を継続するような実行ファイルを作成する。この実行ファイルは、シグナルを受けた時点のプロセスコンテクストを含んだものである。

3.2　継続方法

シグナルによって作成されたファイルを通常のプログラムと同じように実行すると、中断した時点から実行が再開される。また、中断と継続実行は1度だけでなく何回でも可能である。

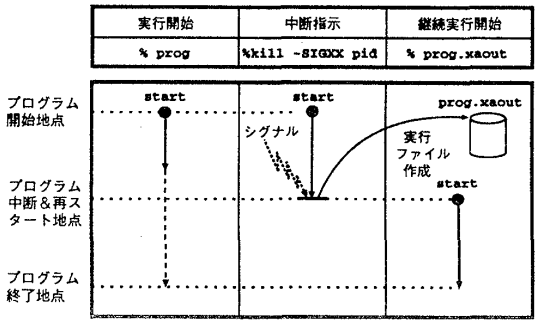

図1：中断・継続実行のイメージ

An Implementation of a Mechanism to Provide Persistent Process on Unix Kernel.
*Naoto Okino
†Yoshiyuki Nakamura
‡Yoshikatsu Tada
§Graduate School of Information Systems, The University of Electro-Communications.

4 実装

4.1 プロセスコンテクストの保存

保存処理の概要

カーネルがシグナル配送の依頼を受けたときは、以下のような処理をする。

- 中断させるプロセスの状態がプロセスコンテクストを保存してもよい状態なら保存する。
- 中断させるプロセスの状態がプロセスコンテクストを保存すべきでない状態なら可能な状態になるまで待って、保存（実行ファイルを作成）する。

保存処理の問題点

プロセスコンテクストの保存処理では、中断されるプロセスの状態を判断して行う必要がある。なぜなら、コンテクストの保存処理をシグナルを受けると同時に行う場合と、プロセス状態の変化を待ってから行う場合とでは、保存するコンテクストや復元処理が大きく異なるからである。たとえば、中断されるプロセスがI/O処理終了待ちのスリープ状態の時に中断してしまうと、継続実行時にもそのような状況を作らなければならない。この場合は復元時に保存するコンテクストが増えたり復元処理が複雑になる。具体的には、コンテクスト保存時にプロセスが待っている要因（この場合、I/O処理）に関するコンテクストを保存する。そして、復元時にはsleepキューに復元するプロセスを登録し、プロセスが中断時に完了を待っていた要因を復元する処理（完了したらプロセスを起こしてくれる処理）を行う必要がある。しかし、I/O処理の終了を待ってから中断すれば、待ちの状況を復元するのに必要なコンテクストや処理が必要なくなる。

4.2 実行ファイル形式

中断時に作成される実行ファイルは独自の形式で、内容は継続実行に必要なプロセスコンテクストである。現在保存しているプロセスコンテクストは、保存するプロセスのメモリイメージ、U領域、proc構造体、中断時のレジスタ群である。

4.3 プロセスコンテクストの復元

復元処理の概要

(1) カーネルがその実行ファイル形式を解釈する。

(2) プロセスコンテクストを取り出す。

(3) 中断時と同じ状態になるようにコンテクストを保存した場所に戻す。

復元処理の問題点

復元するプロセスコンテクストは次のように分類できる。復元時にはそれぞれの分類に応じた復元処理を行なう必要がある。

- 復元しなければならないコンテクスト
（たとえば、メモリイメージ、レジスタ）
- 復元してはいけないコンテクスト
（たとえば、親プロセスのproc構造体のアドレスを格納しているproc構造体の要素: p_pptr）
- 復元時になんらかの変換をするコンテクスト
（たとえば、ウェイトチャネルp_wchanが中断時と再開時で違う場合）

4.4 実験

本研究ではAT互換機上のFreeBSDを拡張して機能を実装した。現在の実装では、保存・復元している主なプロセスコンテクストがプロセスのメモリイメージ、U領域、proc構造体だけなので、プログラムによっては正しく継続実行できない。

拡張したカーネルで、素数を計算して出力するプロセスを中断させ、OSを再起動してから、継続して実行ができることを確認した。

5 おわりに

本研究では、プロセスの永続性を提供する機能を既存のOS上に実装した。この結果、コンピュータを停止してもプロセスはファイルという形で残せるのでプロセスを永続的に扱えるようになった。また、任意の時点でプロセスを中断・継続させようとすると、保存するコンテクストが増え、復元処理が複雑になることを示した。

参考文献

[1] McKusick, M.K., K.Bostic, M.J.Karels and J.S.Quarterman, The Design and Implementation of the 4.4BSD UNIX Operating System, Addison-Wesley, 1996.