プロセスの一時停止と再開を実現する
システムコールの作成に向けて

日大生産工(院)　○山﨑　淳史
日大生産工　松田　聖

1 はじめに

近年、ネットワークの普及により多数のコンピュータが接続された環境を有効に活用する分散処理への取り組みが活発になっている。分散処理では処理を分割して複数のコンピュータで行なう事で高速な処理が期待できる。しかし処理を分担するには、他のコンピュータとの協調が重要になる。例えば一つのコンピュータの処理が中断してしまうと処理全体に影響を与えてしまう。これを回避するには、あるコンピュータが処理を中断する場合、現在実行している処理を一時停止して他のコンピュータに送り、処理を引き渡す必要がある。

本研究では既存のオペレーティングシステム（以下OS）内に分散処理機構の構築を目指している。最初の段階として、実行中の処理の一時停止と再開を、同一システム内で実現する方法について検討する。

2 概要

2.1 環境

本分散処理機構はMINIXというOS上に構築する。MINIXはA・S・タネンバウムによって開発されたUNIX互換のOSである。MINIXはC言語で記述され、ソースコードは全て公開されている。また、自由な変更が許可されている。ネットワークサーバ用OSとして広く使われているUNIXやLinuxよりもOSのサイズが小さく、研究・教育用のOSとして用いられている。

2.2 2つのシステムコール

OS上で実行中のプログラムをプロセスと言う。分散処理において、あるコンピュータが実行中の処理を一時停止して他のコンピュータに引き渡し、処理を再開させるという事は、プロセスをそのままの状態で保存し、他のコンピュータに転送し、そこで保存した状態を復元しプロセスを再開する事である。このプロセスの状態を保存するための「プロセス一時停止システムコール」と、保存した状態を復元しプロセスを再開する「プロセス再開システムコール」の作成を行なう。

次章でまず、プロセスの一時停止・再開を実現する為に必要なプロセスの概念を記す。

3 プロセス

3.1 プロセスの生成・消滅

プロセスの生成と消滅をシェルを例にとって記す。MINIXなどのOSではforkシステムコールでのみプロセスを生成することが可能である。シェルから目的のコマンドを実行する場合、シェルはforkシステムコールを使い、自分の子プロセスを生成する。生成された子プロセスは親プロセスと全く同じ状態を持っている。そこで子プロセスはexecシステムコールを使い目的のコマンドを読み込んで自分自身に置き換える。処理が終了するとプロセスはexitシステムコールを使い、OS上から消滅する。

3.2 プロセス情報

プロセスを実行するためには様々な情報が必要となる。OSは実行中の全てのプロセスを管理しているため、その情報を保持している。プロセスの実行に必要な情報を表1に示す。

表1 プロセスに関する情報

(プロセス管理)	(メモリ管理)	(ファイル管理)
レジスタ	テキストセグメントへのポインタ	UMASKマスク
プログラムカウンタ	データセグメントへのポインタ	ルートディレクトリ
プログラムステータスワード	プロセスグループ	ワーキングディレクトリ
スタックポインタ	実ユーザ識別子	ファイル記述子
プロセス状態	実グループ識別子	実行ユーザ識別子
次回のアラーム時刻	シグナルのビットマップ	実行グループ識別子
シグナル保留ビット	各種フラグビット	システムコールのパラメータ
プロセス識別子	…	各種フラグビット
各種フラグビット		
…		

MINIXでは表1で示した情報が入ったプロセステーブルを構造体配列として保持しそれぞれのプロセスに1つのエントリを与えている。

Toward the Development of System Call for Suspending and Resuming Process.

Atsushi YAMAZAKI and Satoshi MATSUDA

４ プロセス一時停止システムコール

プロセスを一時停止させ、その状態を保存するシステムコールを作成する。

2.2で述べたように、実行中の処理を一時停止するという事は、プロセスをそのままの状態で保存する事である。プロセスはメモリ上に存在し、そのプロセスの情報はプロセステーブルに存在する。よってプロセス一時停止システムコールはメモリ上のデータと該当プロセステーブルの保存を行なう。

プロセスの一時停止は次の手順で行なわれる。

①一時停止して状態を保存したいプロセス（以下、プロセスA）に対してプロセス一時停止システムコールを実行する。

②プロセス一時停止システムコールはプロセステーブルの情報からプロセスAのメモリ空間を参照し、その内容をファイルとして保存する。同時にプロセスAのプロセステーブルの情報もファイルとして保存する。

③全て保存が終了した後、プロセスAに対してkillシステムコールを実行しプロセスAを消滅させる。

①、②、③を行なうことで、プロセスAはプロセスとしてではなく、ファイルとして一時停止させた状態にすることが可能である。

５ プロセス再開システムコール

ファイルとして保存したプロセスの処理を再開させるシステムコールを作成する。

2.2で述べたように、一時停止した処理を再開するという事は、保存したプロセスの状態を復元しプロセスとして処理を再開させる事である。そこでまずforkシステムコールによって適当なプロセスを生成し、そのプロセスに対してプロセス再開システムコールでファイルとして保存したメモリとプロセステーブルの復元を行なう。

以下に4でファイルとして一時停止させたプロセスAを再開させる手順を示す。

①forkシステムコールによってプロセスBを生成する。

②プロセスBが、プロセスAの状態を保存したファイル名をパラメータとしてプロセス再開システムコールを実行する

③、プロセス再開システムコールはファイルからプロセスAのメモリのデータとプロセステーブルのデータを読み込み、プロセスBのメモリ空間とプロセステーブルにそれぞれ書き込む。

①、②、③を行なうことで、プロセスBはプロセスAが保存された（一時停止）された状態と全く同じ状態になり、プロセスAとして処理を再開することが可能となる。

６ まとめ

OS内部に分散処理機構を構築するための、プロセスの一時停止と再開を実現するシステムコールについて提案した。この方法はファイルを使用する場合と、他プロセスとプロセス通信をする場合が考慮されていない。他に今後はこの点に関しての検討が必要である。

「参考文献」

１）A・S・タネンバウム，A・S・ウッドハル，「オペレーティングシステム第2版 設計と理論およびMINIXによる実装」, ピアソン・エデュケーション, (1998)

２）桜田幸嗣，「MINIX C プログラミング」，アスキー，(1993)

３）森田茂男，多田好克，利用者レベルで実現したプロセス移送ライブラリ，情報処理学会研究報告，vol.1991 no.063, (1991)，pp. 41-47