プロセスイメージを用いたLINUXにおける再実行の方法とその問題点について

02G476 舩津 俊一 （最所研究室）

プロセスイメージイメージの履歴を用いたデバッグシステムの研究の一環として、LINIXにおけるプロセス切り替えの機構の理論に基づいたプロセスの再実行の方法とその問題点について検討する。

1 はじめに

近年のソフトウェア開発の現場において、デバッグとテストにかかる時間と費用は増大している。

このような状況に対して、プロセスイメージの履歴を用いた効率的なバグの発見及びテスト手法を提供している[1]。提案のシステムでは、プロセスイメージの履歴を保存しそれを用いてプロセスを復元しデバッグやテストに用いる。本研究では、プロセスイメージから、プロセスの再実行を実現する方法とその問題点について議論する。具体的には、LINUXを対象として、プロセスイメージからプロセスの再実行を実現する方法とその問題点について示す。

プロセスの再実行とは、ある一定のポイントまでプロセスの実行を進め、その状態を保存し、一旦、プロセスの実行を停止させた後、再度、その状態から実行するものである。

しかし、プロセスを再実行するには、プロセスイメージに加え、様々な情報が必要なので、再実行は非常に困難である。このため、本論文では、保存されたプロセスイメージ等からプロセスを再実行するために最低限必要な機能を単純なプロセスを用いて調査し、その後、様々な条件を持つ複雑なプロセスを復元する際の手法とそのとき起こる問題点について議論する。

2 プロセスの再実行に必要な情報

2.1 LINUX内でのプロセス管理情報

LINUXカーネルは、各プロセスの状態（属性、メモリ空間の割り当て等）をプロセスディスクリプタ[2]と呼ばれる構造体で管理している。しかし、プロセスディスクリプタの持つ再実行に必要な情報を、再実行の処理でそのまま用いることができないものが多く含まれている。

そこで、現在実行中のプロセス情報をユーザに与えるためにLINUXはprocファイルシステムという機構を持っている。この情報とプロセスディスクリプタの情報を併用することで再実行ができるかどうか検討する。

2.2 メモリ空間の情報の獲得

LINUXでは、ptraceシステムコールを用いることにより、プロセスのメモリ空間のデータを読み書きできる。しかし、このシステムコールは、対象外プロセスへ影響を与えないように、カーネルメモリ空間へのアクセス禁止など制限が多い。本研究では、この制限をカーネルから削除した。さらに、再実行されるプロセスと再実行させるプロセスを親子関係にすることによりシステムコールの引数の1つであるプロセスID(PID)を容易に獲得できるようにしている。

3 単純なプロセスの再実行

本研究では、1クォンタムプロセスと呼ばれる非常に単純なプロセスの再実行から実現した。1クォンタムプロセスとは、カーネルや他のプロセスに干渉せず、LINUXに割り当てられた1単位時間内で終了するプロセスのことである。

プロセスの再実行の流れを図1に示し、その手順を以下に示す。

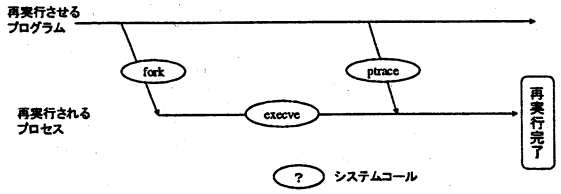

図 1: プロセスの再実行の流れ

1. 再実行されるプロセスをfork,execveシステムコールを用いて生成する。
2. 再実行されるプロセスを休止状態にする。
3. ptraceシステムコールを用いて、メモリ空間とプロセスディスクリプタを書き換える。
4. 休止状態を解く。

execveシステムコールが実行された後に1つの問題が発生する。それは、再実行させるプロセスの方が、先にCPUを獲得する保証がないために、1クォンタムプロセスが実行を終えてしまう問題である。これは

一般的なプロセスの場合も無視できない問題である。この問題に対処するために、本研究では、再実行を行う対象プロセスを起動させる実行ファイルに変更を加え、初期化シーケンス終了時に無限ループする方式とSTOP シグナルを自分に送る方式を試みた。

その後、休止状態になったプロセスに対して、プロセスイメージに基づいて、ptrace システムコールを用いて書き換える。このとき、PID や制御端末のリンクなど元に戻してはならない項目を戻さないようにしなければならない。

4 一般のプロセスの復元の方法の検討

4.1 OPEN 中のファイルに関する処理

プロセスがファイルを使用している場合、使用しているファイルのパス名、ファイルディスクリプタ ID、ファイルへのアクセス位置をプロセスイメージとして保持しておき、再実行の際に復元する必要がある。パス名とファイルディスクリプタ ID は、proc ファイルシステムから知ることが出来る。ファイルへのアクセス位置は、親子プロセスでファイルディスクリプタ ID を共有できることを利用することによって獲得・変更が可能になる [3]。

なお、ファイルの復元は CVS 等のバージョン管理システム [4] と連携することにより可能となる。しかし、これだけでは、LINUX 内にデータがバッファリングされている場合に問題が残る。

4.2 ページテーブルの変化を伴うプロセス

動的メモリ管理を行っているプロセスの場合、ページテーブルが変化する。このようなプロセスを再実行する際には、メモリマップおよびページフレームの整合性を取る必要がある。そうしなければ、ページフォルトやセグメンテーションエラーを起こす。そして、メモリ空間のセグメントのサイズ変更を行う brk システムコールでは、自分のセグメントセットのヒープ領域にしか対応しない。また、mmap システムコールの場合、任意のアドレスを確保することが出来るが、共有セグメントを破壊する可能性を秘めている。

4.3 共有ライブラリを使用するプロセス

プロセスが静的ライブラリではなく、共有ライブラリを使用していた場合、ダイナミックリンカがアドレス解決を行う。そして、そのリンク先は、共有セグメントである。共有セグメントの取り扱いは難しく、ページフレーム共有を処理する機構が必要であるので実現は困難である。

4.4 システムコール返り値消失問題

プロセスが、システムコールを呼び出し、その返り値を待っている状態でプロセスイメージを取ると、その再実行の際に、通常のプロセス切り替えを用いた本研究の再実行の方法では、返り値がきちんと設定されないという問題が発生する。この問題解決には、カーネルが持つプロセスイメージも必要となる。

4.5 2つ以上のプロセスが関わるプログラム

他のプロセスと通信を行い、同期を取りながら動作するプロセスの場合、LINUX ではパイプやソケットは、ファイルと同等の扱いが出来るので、復元が可能である。しかし、複数のプロセスで同期を取りながらプロセスイメージの保存や復元する必要であるので非常に難しく、タイムアウトや接続先消失などの問題が発生する。

5 まとめ

本論文では、卒業研究 [1] の続きとして、プロセスイメージ履歴管理システムのプロセスイメージの利用法の 1 つであるプロセスの再実行について示した。

LINUX が持つプロセス切り替えの機構を利用して、メモリ空間およびプロセスディスクリプタを書き換えるという手段を用いることにより、再実行が可能であることを確認した。さらに非常に限定しているが、1 クォンタムプロセスと呼ばれる単純なプロセスの再実行に成功した。

しかし、実際のプロセスは、ファイルやライブラリを使用していたり、動的メモリ管理を用いていたりする。そのような、複雑なプロセスを復元するためには、さらに必要となる新たな理論や技術について検討しなければならない。

参考文献

[1] 舩津 俊一.“プロセスの実行イメージの履歴の獲得についての研究”,
香川大学工学部信頼性情報システム工学科卒業論文,2002.

[2] DANIEL P.BOVET and MARCO CESATI (高橋浩和 監訳. 杉田由美子, 高杉昌督, 畑崎恵介, 平松雅巳, 安井隆宏 訳).
“詳解 LINUX カーネル 第2版”,
O'REILLY (オーラリージャパン),2003

[3] 塚越一雄. “Linux システムコール”, 技術評論社”, 2000.

[4] 鯉江英隆, 西本卓也, 馬場肇.
“バージョン管理システム（CVS）の導入と活用”,
ソフトバンク パブリッシング, 2000.