プロセスの実行イメージの履歴の利用法についての研究
-Windows 上におけるデバッグ情報の獲得-

00T228　久保敦史(最所研究室)

本研究では、プロセスの実行イメージの履歴を用いたデバッギング手法の有用性を調べるために、実行中のプロセスに対し、デバッグ情報を取り出す事を試み、レジスタ情報とスタックフレームの情報を取り出す事が出来た。その結果、プロセスイメージに対しても同様の情報を取り出す事ができれば、効果的なデバッグが可能である事が分かった。

1　序論

何らかの作業中にコンピュータが意図しない停止を起こしたとすると、それまでの作業は失われてしまう。こうした損失を防ぐために、停止した状態の復元を目指して、我々の研究室ではプロセスの実行イメージの履歴を保存し、再現する研究を行っている[1]。また、ソフトウェア開発の観点から見ても、デバッグやテストに割く時間は相当なものであり、効率的なデバッギング手法が重要なものとなっている。そのため、我々の研究室ではプロセスの実行イメージの履歴を用いたデバッグシステムの構築を行っている。本研究では Windows を対象に、履歴を用いたデバッギング手法の有効性について調査する。具体的には、実行中のプロセスからレジスタ情報やスタックフレームの情報を取り出すツールを作成し、そのツールを用いて取り出した情報がデバッグを行う上で有効であるかを調査する。

2　概要

プロセスイメージの履歴を用いたデバッグシステムの全体の構成を図 1 に示す。このシステムでは過去におけるプロセスの実行イメージの履歴を保存しておき、バグが発生した際にこの履歴を利用してデバッグを行っていく。まず、デバッガ部はシステムに対してプロセスイメージのバージョンを指定して要求する。これに対して、システムは要求されたバージョンのプロセスイメージをデバッガに引き渡す。このプロセスイメージに対してデバッグを行う。この状態でバグが見つからなければ別のバージョンを指定して再度プロセスイメージを要求し、同様の手順を踏んでデバッグを行う。これを繰り返す事で、バグを発見する可能性が向上する。

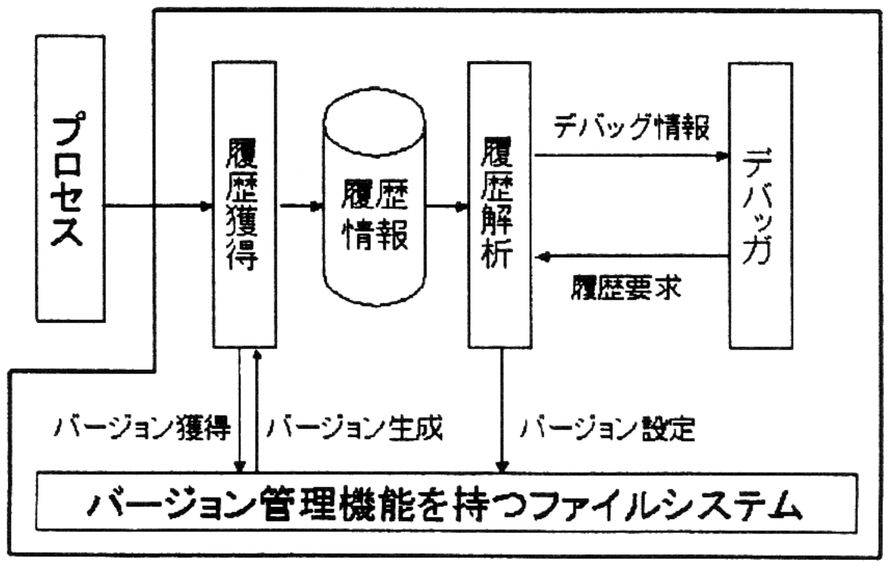

図 1:デバッグシステムの概要

本研究では、デバッグを補助する情報として、プロセスの情報を取り出すツールの構築を行う。従来のデバッガでは、現在の状態から以前の状態に遡って調べる事は出来ないため、こうした履歴から得られる情報は有益であると思われる。デバッグを行うにあたって有益と考えられる情報は、レジスタ情報、スタックフレームの構成、変数情報である。なお、これらの情報は従来のデバッグシステムでもよく利用されているものである。

3　設計

3.1　プロセス情報の取得

デバッグの補助となる情報を抽出するにあたって、まず、メモリ空間の取得が必要になる。Windows においては ReadProcessMemory 関数を用いる事で、任意のプロセスの、任意のメモリ空間を取得する事が可能である。ただし、この関数によって得られるデータは純粋なバイナリデータであるため、解析が必要となる。

スレッドのレジスタ情報はコンテキストの一部として取得する事が出来る。コンテキストとは、ユーザ空間、あるいはカーネル内のどちらかで動作しているスレッドが、コンピュータのレジスタ内に保持している値の集合である。取得出来るレジスタ情報のうち、スタックフレームの構成と密接な関係にある、プログラムカウンタ(EIP)、スタックポインタレジスタ(ESP)、フレームポインタレジスタ(EBP)は、本研究において特に重要である。これらを用いて、現在、どの関数から呼ばれた、どの関数が実行されているか等の情報が分かる。

3.2　プロセス情報の解析

図 2 に示すように、一般に、スタックは ESP と EBP の 2 つによって制御されている。EBP を使わずに ESP だけを用いるプロセスも存在するが、そのプロセスに対してはスタックフレームの追跡は困難である。ここでは EBP を利用した、スタックフレームの追跡が可能なプロセスについて考えていく。

Windows では、スタックトレースを行うための汎用的な関数、StackWalk[2]を提供している。この関数により、スタックフレームの情報を順に調べていく事が出来る。StackWalk 関数によって得られるスタックフレーム情報には、プログラムカウンタ、フレームポインタ、スタックポインタの情報が含まれており、この値から、スタックフレームが指し示す関数の名前や位置を取得出来る。

変数情報を取得するためにはシンボルテーブルが必要となる。しかし、現時点ではシンボルテーブルの取得が出来ておらず、変数情報を得る事は出来ていない。

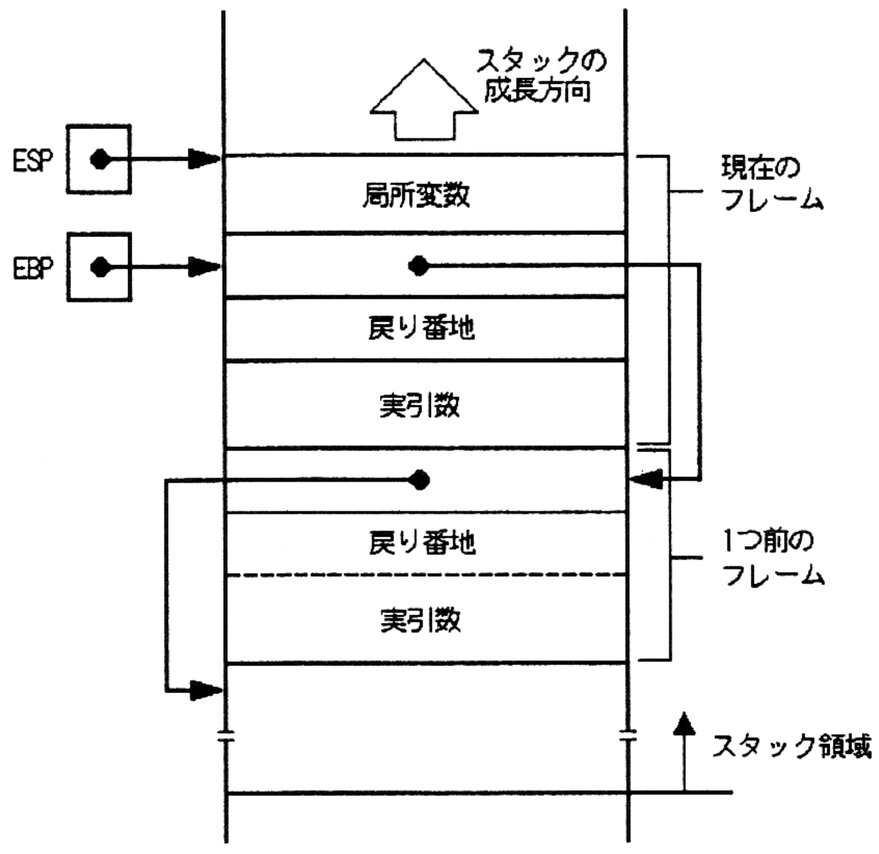

図 2:スタックフレームの概念図

4 実装

本ツールの開発環境として、OS は WindowsXP、言語は Visual C++を用いた。実装した機能を以下に示す。

- 実行中のプロセスの一覧の取得と表示
- 任意のプロセスのモジュール情報とヒープ ID、スレッド ID の表示
- レジスタ情報の表示
- 任意のモジュールのメモリ空間の保存
- スタックフレームの表示
- プロセス情報の保存

なお、本ツールは実行中のプロセスの情報を取得するものである。したがって、任意の地点の情報を取得するためには、何らかの手段によってプロセスを一時的に停止させなければならない。本研究では、Visual Studio .NET2003 のデバッガを用いてプログラムを一時的に停止させる事にした。

本ツールにより取得出来る情報の例を表 1、表 2、表 3 に示す。

表 1:モジュールのメモリ情報

```
sample.exe
0x00400000 - 0x0042AFFF
4D5A90000300000004000000FFFF0000
B800000000000000400000000000000
0000000000000000000000000000000
0000000000000000000000D0000000
0E1FBA0E00B409CD21B8014CCD215468
69732070726F6772616D2063616E6E6F
```

表 2:レジスタ情報

EAX	0x00000014
EBX	0x7FFDF000
ECX	0x00000000
EDX	0x00000001
ESI	0x00000000
EDI	0x0012FDE8
EIP	0x00411A69
ESP	0x0012FD10
EBP	0x0012FDE8
EFL	0x00000212

表 3:スタックフレーム情報

```
0x00411A69 sample.exe
mul +41 byte(s)

0x00413109 sample.exe
main +57 byte(s)

0x00411E30 sample.exe
mainCRTStartup
+368 byte(s)

0x77E414C7 kernel32.dll
GetCurrentDirectoryW
+68 byte(s)
```

表 1 は本ツールで取得出来るメモリ情報である。1 行目がモジュール名、2 行目が保存したメモリアドレスの範囲、3 行目以降がメモリ情報となっている。表 2 はレジスタ情報である。この中で、EIP、ESP、EBP の値はスタックフレーム情報を調べる時に使用する。表 3 はスタックフレーム情報である。2 行で 1 つのスタックフレームを示し、順にプログラムカウンタが指すアドレス、そのアドレスを含むモジュールの名前、関数名及び関数の先頭からの位置を表している。現在実行中の関数から順に、呼び出し元の関数を表示する。表 3 に示す例では、現在実行中の地点が mul 関数の先頭から 41bytes の位置である事、main 関数から 57bytes の位置で mul 関数が呼び出された事等、プロセス全体の流れを把握する事が出来る。

5 まとめ

本論文では、プロセスの実行イメージの履歴を用いたデバッグシステムの有効性を調べるために、Windows 上におけるメモリ空間の獲得と、実行中のプロセスを対象に、デバッグ情報を抽出する方法を検討した。その結果、実行中のプロセスに対してはデバッグ情報を取得する事が出来た。本ツールと同様の手法をプロセスイメージに対して適用出来れば、取得したプロセスイメージに対してもデバッグ情報を取り出す事が可能となる。

参考文献

[1] 船津俊一, “プロセスの実行イメージの履歴の獲得についての研究”, 香川大学工学部卒業論文集, 2001

[2] John Robbins 著, 豊田孝 訳, “.NET & Windows プログラマのためのデバッグテクニック徹底解説”, 日経 BP ソフトプレス, 2003