2003－OS－93 (20)
2003／5／9

ファイルI/Oと後戻りを両立させたデバッガの作成

吉田 悟† 梅村恭司†

デバッグにおいて，実際に動いているマシンの上でプログラムのデバッグを行うと，ファイルI/Oなどのマシンの状態について記憶しておくことが難しいため，ファイルI/Oを使用するプログラムなどについては後戻り可能なデバッガを作ることが困難である．そこで我々は，仮想マシン上でプログラムを動かし，状態の保存の際にファイルの状態なども仮想マシンの状態として保存することによって，より高度なデバッグ環境を提供する．本研究では，仮想マシンとしてPalmSource.IncのPalm OS Emulator(POSE)を用い，後戻り可能なデバッガを実現した．

A debugger with both roll back mechanizm and file operations

Satoru Yoshida† and Kyouji Umemura†

In debugging, it is difficult to memorize all the state of machine, especially file operation state. When a program with file operation is debugged, a debugger that will roll back the state of the program is difficult to realize. We propose more advanced debugging environment by moving a program on a virtual machine and saving the state of the program including file status. With this debugger, both rolling back and file operation is possible, using Palm OS Emulator (POSE) of PalmSource.Inc as a virtual machine.

1. はじめに

プログラミングとバグは切っても切れない関係があり，バグを発見するためのツールであるデバッガの機能が向上すれば，プログラムの開発効率が上がると考えられる．一般的なデバッガには，プログラム実行中に動的に変化する変数やCPUのレジスタなどを参照したり，それらの値を保存して，プログラムをその時点から再実行するような機能などがある．しかし，これらの一般的なデバッガでは，I/Oに関連するデバイスへのアクセス，たとえばハードディスクなどを書き込んだ後などでは，書き込み前に戻すことは難しい．このようなデバッグにおいて重要な情報であるが，その保存が難しいものを保存しようというのが本研究の目的である．

本研究では，これらの保存が難しい情報を保存するため，実在のマシン上でデバッガを動かす方法を取らず，仮想マシン上でデバッガを動かし，I/Oを含めた各種情報を保存可能である閉じたI/Oを作ることによってマシンの多くの情報を保存している．たとえば，ハードディスクの書き込みを行った場合，書き込みを行う前のディスク全体のイメージを取っておけば，書き込みをする前に戻る事ができる．このように，仮想マシンを使用することによってマシンのI/Oの多くが保存可能となるので，高度なデバッグ環境を提供することができる．

本研究では，このデバッグ環境の構築するためのプラットフォームとしてPalm OS Emulator(POSE)を選択した．POSEはPalm OSというPDA用のOSのエミュレーションを行うことができ，GNUライセンスでソースコードが配布されており，自由に改変，配布ができる．本研究では，POSEのソースコードに変更を加え，各種デバッグ情報を保存でき，再実行できるシステムを作成した．

2. POSEについて

ターゲットOSであるPalm OSと，ターゲットマシンであるPOSEについて説明する．

Palmは，Graffitiと呼ばれる手書き入力部とタッチパッド，いくつかのボタンがある手のひらサイズのマシンである．Palm OSは，Palmでこれらの入力装置を使ってGUIで入力を行うことができるPDA(Personal Digital(Data) Assistant)用のOSである．PDA用であるため，Windowsなどに比べて動作に必要なCPUの処理能力は低く，主記憶，仮想記憶ともに少ない容

† 豊橋技術科学大学情報工学系
Department of Computer and Information Sciences, Toyohashi University of Technology

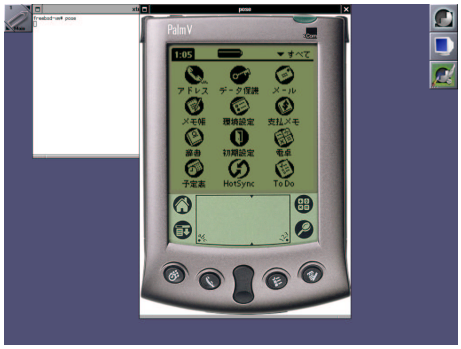

図 1 POSE
Fig. 1 POSE

量で実装されている．また，無料で手に入るソフトウェアの開発環境として，POSE と呼ばれるエミュレータが提供されている．その他の特徴として，一般的な PC で言うところのハードディスクとメモリが RAM のみで代用されている点や，プログラミングはイベント駆動で行う点などがある．

POSE は，Palm のエミュレータである (図 1)．ソースコードは GNU ライセンスにて配布されており，自由に改変，配布が可能である．本研究では，POSE のソースコードに手を加え，目的の機能を実装した．

3. システムの概要

今回作成したシステムの概要を説明する．

POSE にはセーブ機能があり，この機能は，POSE がエミュレーションを行っている CPU のレジスタや RAM，タッチパッドのどこをポインティングしているかなどの現在の状態を保存する．しかしセーブは人手で行わなくてはならず，デバッグに用いるには実用的ではない．そのため，今回作成したシステムでは，一定時間経つごとに POSE のセーブ機能を自動で実行 (オートセーブ) し，その状態を保存するという方法で各種デバッグ情報を保存している．これにより，オートセーブされた時間がわかれば，その状態を POSE のロード機能でロードすればその時点までもどることができる．これに加え，本システムでは，一つ前の状態に戻る機能 (Undo)，一つ先の状態に進む機能 (Redo) を追加し，簡単に状態のロードができるようになっている．

4. システム実現の留意点

今回作成したシステムの実現に当たっての留意点を説明する

POSE は GUI で動くイベント駆動型のプログラムであり，POSE の最下位では，各種イベントを待つイベントループが実行されている．オートセーブを行うにあたり，本システムではそのイベントループ中に一定時間でセーブを行うように変更を加えた．ただし，セーブを行うには POSE の動作を一時的に止める必要があり，セーブを行う前に POSE を停止させるようにした．また，POSE の状態のロードには，状態をセーブしたファイルのフルパスが必要である．一つ前の状態に戻る機能 (Undo)，一つ先の状態に進む機能 (Redo) は，ソースにファイルのフルパスを保存する双方向の循環リストを追加することにより実現している．

今回作成したシステムの実行の様子を図 2，図 3 に示す．図 2 では，画面の中央から時計回りに連続して線を描いていくプログラムを実行している． 図 2 の A のような状態でオートセーブがされたとする．この状態からさらにプログラムの実行を続けていき，図 2 の B のような状態になったときに Undo を呼び出す．すると図 2 の C の状態となる．これは図 2 の A でセーブされた状態であり，つまりは後戻りが行われる．Undo が呼び出された直後，プログラムはその時点から再実行され，図 2 の C の状態から再び線が描かれていく．また，Palm の標準アプリケーションであるメモ帳が後戻りする様子を図 3 に示す．図 3 の A の状態では，メモ帳に何も保存されていない．この状態でオートセーブが行われたとする．そして図 3 の B のようにいくつか新規でデータを作成し，保存する．そして Undo を呼び出すと図 3 の状態となり，これは図 3 の A の状態となる．このように，本研究で作成したしシステムは，ファイル I/O が行われても後戻りができる．POSE は一般的な PC におけるハードディスクとメモリにあたる RAM の大きさを起動時に指定できる．RAM の容量とセーブに要する時間，およびオートセーブ時に POSE が保存したデータのサイズの関係を表 1 に示す．なお，すべての RAM サイズにおいて，起動直後に図 2 で実行されているプログラムをインストール，実行し，8 回セーブを行ったときの平均を取ったものである．圧縮率低とかかれているものは，これに加え辞書ファイルをほぼ RAM の容量となるようにインストールしたものである．エミュレータを動かすマシンの CPU は AthlonXP2100+(1.73GHz) を用い，ハードディスクの書き込み速度は 40MB/s 程度のもの (C によるプログラムを作成して計測した) を用いた．表 1 からわかるとおり，RAM のサイズと実際に保存されているデータのサイズは大きく異なる．これは POSE がセーブの際 RAM のイメージを圧縮しているためである．上の 4 つに関しては圧縮率が

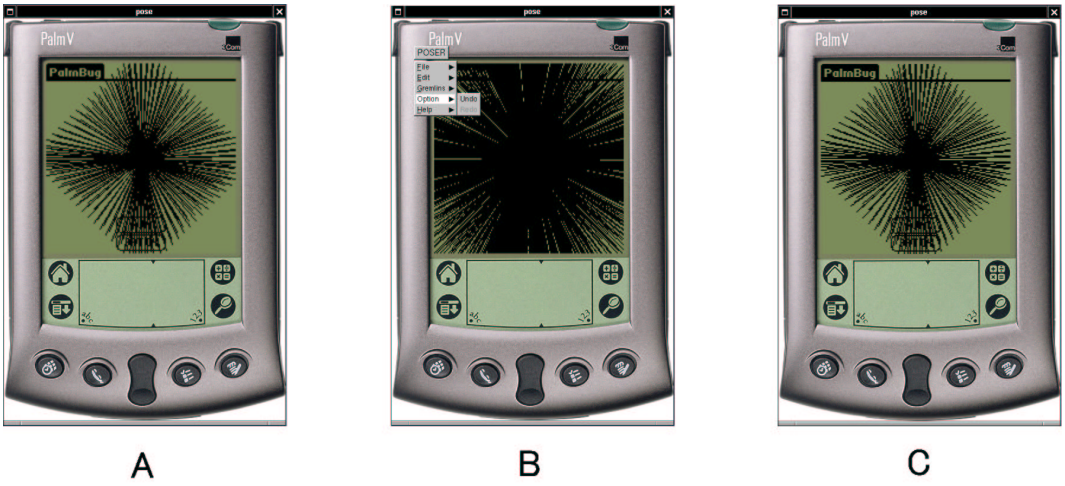

図 2 自作プログラムでの Undo の動作
Fig. 2 Operation of Undo in a original work program

高いが，これは起動直後で RAM イメージが何らかの方法 (ゼロクリアなど) で初期化されており，かつ，POSE でデフォルトで使用される RAM は OS が確保する 96KB ほどであるためである．よって RAM の容量が 8192KB でも，実際にはほとんど使用していないため，圧縮されたデータのサイズはずいぶん小さいものとなっている．表 1 からもわかるとおり，圧縮率が低くなるとセーブに要する時間は大きく増大することになる．POSE は C++で実装されており，ハードディスクの転送速度に対してセーブ処理の速度低下が大きいことは，圧縮率が低くなったときに，POSE の RAM をセーブする処理がボトルネックになっていると考えられる．

本システムのセーブ中は POSE 自体が止まってしまい，I/O が不可能となるのでデバッグ環境としては問題がある．表 1 から，セーブにかかる時間が 300ms 以下であればデバッグ環境として問題が無いとするなら，本システムの RAM のサイズはこの計算機では 1MB 程度までが妥当な容量と考えられる．実際に，Palm のアプリケーションは辞書などの巨大なデータを扱うものでなければまず 1MB 以下となるため，Palm のデバッグ環境としては RAM サイズが 1MB 程度でもデバッグには十分用いることができると思われる．

5. OS の選択と必要な計算機

今回作成したシステムで保存すべき内容は，最大でも 8MB 程度の RAM 領域，CPU のレジスタ，ポインティングしている座標などであり，保存すべきデータはたいして大きいものではなく，合計しても数 MB 程度である．現在，システムを作成したマシンでオートセーブが行われると，POSE は最悪で 2 秒ほど止まってしまう．止まっているときはタッチパッドなどの各種 I/O は使用できないため，デバッグ環境としては問題があるが，これはマシンの性能向上によって解決できるレベルの処理時間である．しかし，Windows などの巨大な OS で，本研究で実装したようなシステムを構築しようとした場合には，保存すべきデータの量は数百 MB 単位になってしまい，各種状態のセーブは可能になったとしても，セーブに時間がかかりすぎて，デバッグ環境としては問題となる．また，保存すべきデータだけでなく，処理速度も問題である．本システムではエミュレートを行う対象となる OS(本システムでは Palm OS) と同時にエミュレータ本体を動かす OS が動いている必要があり，そのため CPU に多大な処理能力が要求される．処理能力が足りなければ，エミュレータの処理速度も遅くなり，I/O などに遅延が発生してしまうことになる．本研究で使用した Palm OS は Dragonball EZ 20MHz で動いていたものであり，これは PentiumIII600MHz のマシン上で遅延などはあまり感じることなく動いていた．マシンをエミュレートするには，エミュレートする実機に対して処理速度が高速な CPU が必要になる．

つまり現状では，このアプローチは保存すべきデータが数 MB 以下となり，処理速度も低速で済む組込用途の OS などの小さなシステムに適応可能である．組込用の OS は，携帯電話や電子レンジ，冷蔵庫など様々な電子機器で用いられており，今回作成したシステムのようなアプローチをとれば，デバッグ環境は大

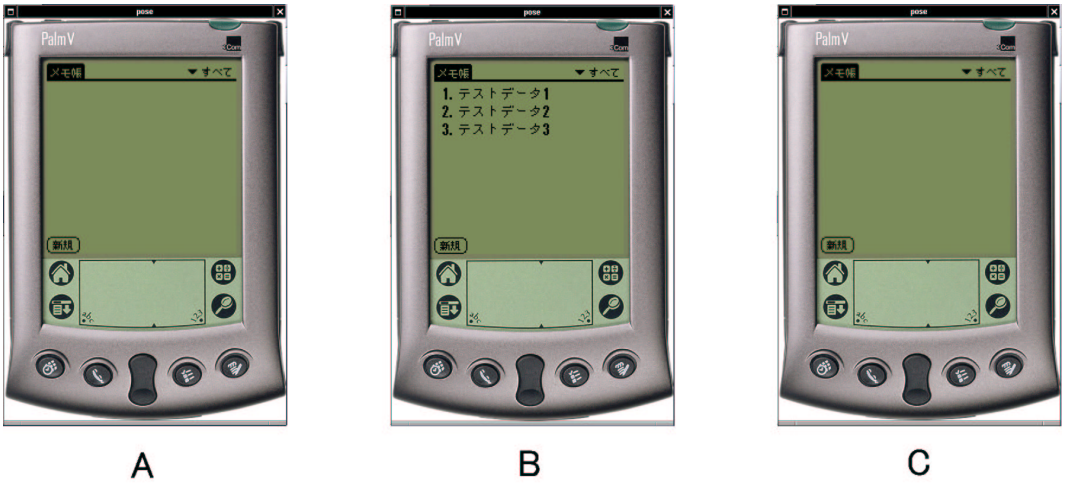

図 3 標準アプリケーションでの Undo の動作
Fig. 3 Operation of Undo in a standard application

POSE の RAM の容量	セーブに要した時間	POSE が保存したデータのサイズ
128KB	88ms	52KB
512KB	103ms	57KB
2048KB	164ms	74KB
8196KB	400ms	129KB
2048KB(圧縮率低)	357ms	1821KB
4096KB(圧縮率低)	701ms	3885KB
8196KB(圧縮率低)	1302ms	7221KB

表 1 POSE のセーブに要する時間とデータサイズ
Table 1 Time and data size which save of POSE takes

きく向上することが予想される．

また，将来的には，Windows などの OS でもこのアプローチで有効なデバッグ環境を構築できるはずである．現在の HDD の転送速度は最大で 85MB/s 程度であり，POSE の RAM のセーブ処理の遅さを無視すると，これは POSE でセーブすべき最大でも 8MB のデータからみれば十分に高速である．しかし，数百 MB のデータを保存しなければならない Windows などの OS からみれば不十分である．セーブにかかる時間が 0.1 秒程度としても，Windows などの OS では数 GB/s の転送速度を持った HDD が必要となる．CPU に関しても，エミュレートする実機に対して必要なクロックが 30 倍であると仮定するなら，200MHz で快適に動く OS をエミュレートするには，6GHz の CPU が必要となる．このような問題を回避するために，OS の持つ仮想記憶メカニズムを利用することを計画している．

6. ま と め

今回作成したシステムは，マシンの I/O を含む各種情報を保存でき，これにより高度なデバッグができるようになった．実際にはデバッグだけではなく，完全に過去に戻ることのできるシステムであり，デバッグ以外の役にも立つ．たとえば，間違えてファイルを消してしまったときなどでも，今回のシステムを使用すれば，ファイルを削除する前の RAM のイメージを書き戻すことにより，ファイルの復帰ができる．過去に戻れるこのシステムは，携帯電話，家電製品などの組み込み OS 上でのプログラミングや，ゲーム機でのゲーム開発などの分野での応用が期待できる．

参 考 文 献

1) ジョナサン・B. ローゼンバーグ : デバッガの理論と実装，アスキー出版局 (1998)
2) PalmSource,Inc. : Palm OS Emulator Excerpt from Palm OS programming Development Tools Guide, PalmSource,Inc,(2002)
3) 3Com : Palm OS バイブル，日経 BP 社 (1999)