プログラムの動的監視システム DONATS の設計と実現

5S－3

西見泰浩, 浅川忠彦, 板野肯三
筑波大学 電子・情報工学系

1 はじめに

他人の書いたプログラムを再利用する時に、まず必要なのがそのプログラムに何が書かれているかを理解することである。この人間がプログラムを解析しようとする作業をコンピュータで支援するために、実行中のプログラムの動きを観察する（動的監視）ことのできるシステム DONATS を作成することにした [1]。

2 プログラムの動的監視

動的監視をするにあたっては、対象となるプログラムに情報を発生させるためのプローブをソースコード中に埋め込む。これをコンパイルし実行すると、プローブを入れた部分に実行の制御がきた順に情報が送り出される。プローブによってとり出された情報は、表示を専門に行うプログラムに送られ表示される。このプログラムはデータの選別や表示の仕方など、ユーザが必要な処理に合わせて作成する。

ユーザは表示を行うプログラムの出力結果を見て、動作の確認を行う。そしてユーザは次の解析に向けて、新たなプローブの挿入や情報の処理方法の変更、表示を行うプログラムのカスタマイズを行う。

3 DONATS システムの実現

本方式を実際の解析に使用して評価するために、動的監視のシステム DONATS を UNIX 上で実現した。

3.1 プローブ

プローブとプローブからのメッセージを受け取るライブラリ関数について説明する。

非同期プローブ: このプローブは、printf 文と似た形式の引数を取る関数であり、引数で指定された情報をメッセージとして送出する。

同期プローブ: このプローブは非同期プローブと同じ引数を取る関数である。非同期プローブとの違いは、メッセージを送出した後に、プログラムの実行を一時停止し、同期を取る返事のメッセージを受け取ることで実行が再開される。

メッセージ受信関数: これはプローブからのメッセージを受け取る関数で、引数として渡される文字列のバッファに受け取ったメッセージが入れられる。

同期メッセージ送信関数: この関数は同期プローブに対して同期のメッセージを返すための関数である。

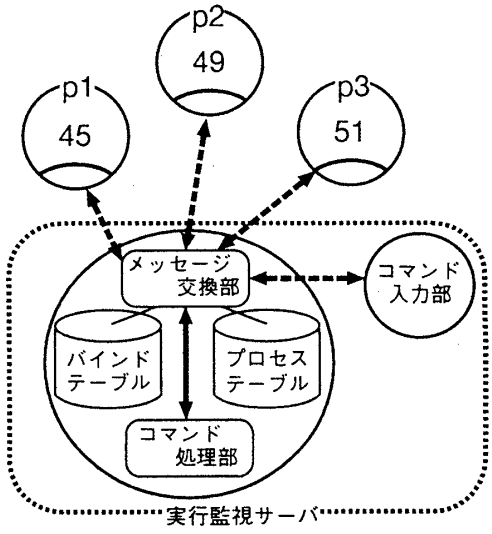

図 1: 実行監視サーバ

3.2 実行監視サーバ

DONATS で実行されるプログラムは、プロセスとして操作される。そして論理的には、DONATS 上のプロセスはメッセージの送受信によるネットワークを構成する。また、DONATS の実現は UNIX の上で行うことにしたので、DONATS 上のプロセスは、そのまま UNIX のプロセスで実現することにした。ここでは DONATS の上で操作するプロセスのことを donats-プロセスと呼ぶことにする。donats-プロセス間のメッセージ通信は、ソケットを使って実行監視サーバを経由して行うことにし、この実行監視サーバ内で、メッセージの分配の制御を行う。実行監視サーバは、ユーザからのコマンドを対話的に処理して、donats-プロセ

Design and Implementation of the Dynamic Program Monitoring System DONATS
Yasuhiro Nishmi, Tadahiko Asakawa, and Kozo Itano
Institute of Information Sciences and Electronics, University of Tsukuba

スの生成や管理を行う機能も持つ。このプロセスの管理は実行監視サーバ中のプロセステーブルで行う。また、実行監視サーバ中のバインドテーブルは、メッセージの送り先を定義したプロセスネットワークを管理するテーブルである。

実行監視サーバの実現として、複数のプロセスからのメッセージ待ちをソケットの select を使って実現しているので、コマンド入力のためにブロックすることができない。このために、実行監視サーバ内の、対話処理を行う部分を、コマンド入力プロセスとして分離し、実行監視サーバの本体プロセスとの間もソケットで接続することにした（図 1)。

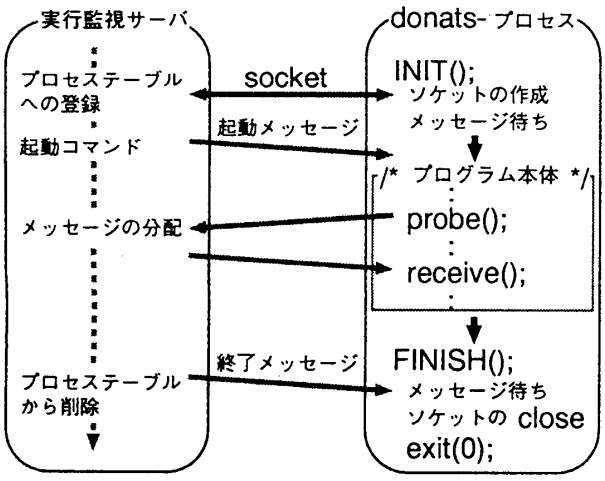

図 2: メッセージ通信モデル

3.3 メッセージ通信モデル

donats-プロセスは実行監視サーバとの間でソケットを使って通信を行うので、プロセスの実行の最初にソケット関係の初期化が必要である。この初期化のために INIT() という関数が用意されている。この INIT() の中では、ソケットの作成後、サーバからの起動メッセージを待ってプロセス本体の実行に入る。また、終了用の関数として FINISH() もあり、これはサーバからの終了メッセージを待ってソケットのクローズとプロセスの終了を行う。このようにプロセスの実行の最初で同期を取るのは、プロセスの作成後にメッセージの送り先の定義を行うためであり、最後で同期を取るのは、プロセスの実行終了したときにその状態を保持しておくためである。

また、メッセージには、プローブによって非同期と同期の 2 つのタイプがあり、これに同期プローブに対する返事のメッセージを加えた、3 つの種類が donats-プロセスの間で送受信される。これはメッセージの先頭に付けられるメッセージタイプ (async,sync,ack) によって処理が分かれる。

メッセージタイプが sync の時は、プローブからのメッセージが送り先のプロセスに送られた後、返事が返ってくるまでもとのプロセスは待つために、sync 型のメッセージには送信側のプロセスの pid が付加される。返事のメッセージはこの pid を付加してメッセージタイプを ack で返送する。

4 DONATS のバージョン管理機能

DONATS を用いて、プログラムの解析を進めていく過程で、過去の解析の内容の保存・参照が可能ならばプログラムの理解がよりスムースに進むと考えられる。具体的に、保存の対象は、解析中のロードテーブル、バインドテーブルの内容、実際にロードされている各プロセスの UNIX 上での実行ファイル、それらを生成するためのソースファイルである。また、参照する場合に、自動的に保存された内容を呼び出し、実行待機状態にする機能を持たせた。

DONATS のバージョン管理機能は現在のところ 2 つのコマンドから構成されている。状態の保存は、save コマンドを用いて行う。このコマンドは、カレントディレクトリの下に、指定されたバージョン名と同名のディレクトリ、バージョン名に拡張子.nats を付加したファイルを作成する。作成されたディレクトリには各プロセスの UNIX 上での実行形式ファイル、それらを生成するソースファイルが一括して保存される。また、.nats ファイルには、ロードテーブル、バインドテーブルの内容が保存される。save コマンドで保存された内容を呼び出すには、recall コマンドを用いる。

5 おわりに

DONATS システムの具体的な実現は、SparcSS1 (SunOS 4.1.3) と Gateway2000 (BSD/OS 2.0) 上で行った。今後は様々な解析例を集め表示プロセスの機能を充実させるとともに、実際にプログラムの解析にこのシステムを使用し評価を行う予定である。

参考文献

[1] 浅川忠彦, 西見泰浩, 板野肯三: プログラムの理解を支援する動的監視システム DONATS, 情報処理学会 第 37 回プログラミングシンポジウム 報告集, pp.109-120, 1996.