4F−12

CHILLセルフ処理系におけるデバッグ機構の検討*

佐々木秀紀†　佐藤規男†
NTTソフトウェア研究所‡

1 はじめに

Sun3上で動作するCHILL[1]の処理系（以後「セルフ処理系」と表記する）が開発されている[2][3]。CHILLは交換処理に適した言語であるが、セルフ処理系は教育用および単体試験用としての需要がある。

セルフ処理系においてはデバッガc.dbが開発され[4]使用されている。本稿では、c.dbとは別に、dbxやgdbなどの既存のC言語用デバッガとCHILL実行時ライブラリ(以後"CRTL"と表記する)内のデバッグ機構を組み合わせたデバッグを提案する。

c.dbは従来のソースレベルデバッガの機能に加えて、並列CHILLプロセス(以後、単にプロセスという場合はCHILLプロセスを指すこととする)群のデバッグ機能を備えている。これは、個々のプロセスに着目したデバッグには効果的であるが、プロセス間の通信に着目したデバッグの機能はない。本稿で提案するデバッグ機構においては、この機能が容易に実現できることを示す。

2 CHILLプロセスと実行時ライブラリ

2.1 プロセスの定義

CHILLにおけるプロセスとは、一度に実行される操作列のことであり、並列処理とは複数のプロセスの実行がオーバーラップしていることである。即ち、あるプロセスの操作が、他のプロセスの操作の終了以前に開始されれば、それらは並列実行されているという。

2.2 CRTLにおけるCHILLプロセス

CHILLではプロセス間で多くの変数を共有するのが一般的である。よってCHILLプロセス1つ1つに対応してUNIXプロセスを生成するのは、共有メモリを明示的に宣言する必要があり、非効率的である。このためセルフ処理系におけるCHILLプロセス群は1つのUNIXプロセス内に生成しており、UNIXプロセスはCHILLプログラムとなる。

現在のセルフ処理系では、CRTL内に独自のスレッドとディスパッチャを生成している。CRTL内に各プロセスのコンテクストを保持しているため、並列プロセスのデバッグのための機構をCRTL内に実現すれば、デバッガ本体に既存のC言語用のデバッガを使用して、実用的なデバッガを作成することが可能であると考えた。図1に全体構成を示す。

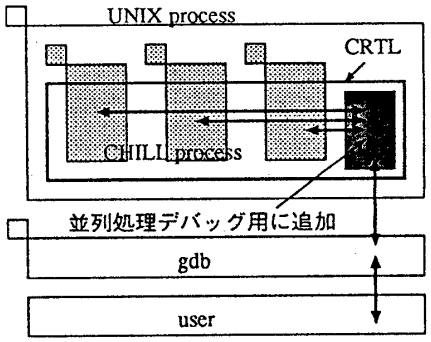

図1: セルフ処理系におけるデバッグ機構

デバッガにはgdb(Ver4.2)をソースの変更をせずに用いることとしたが、関数コールの機能を持つC言語用のデバッガであれば、本稿で提案するデバッグ機構の構築に利用可能である。図1は、逐次プログラムのデバッグにはgdbで対処し、CHILL特有の並列処理部のデバッグに対応するものをCRTLに追加した構成となっている。

3 CHILLにおけるシグナル送受信

プロセス間の通信に着目したデバッガを作成するにあたり、第一段階として、プロセスの同期および通信を実現する機能の一つであるシグナルをとり上げる。CHILLのシグナルは、UNIXのシグナルやセマフォのシグナルとは異なる。以後、単にシグナルという場合はCHILLのシグナルを指すこととする。シグナルのセマンティックスを以下に示す。

シグナル： プロセスの同期と通信を実現する機構の一つ

シグナルの送信： 指定したシグナルが値リストと共ともに受信側プロセスに直接転送される。シグナルを待っているプロセスがなくとも、値はプロセスがそれを受け取るまで保持される。

シグナルの受信： シグナルネームで指定された値リストとシグナルを受信する。該当するシグナルがなければプロセスはdelay状態になる。複数のシグナルが受信可能であれば、任意の一つのシグナルが即時に受信される。

セルフ処理系におけるシグナルの送受信の一例を図2に示す。図2は、プロセスp1とp2が通信しながら並行動作をしている例である。時刻aにp1が送信したシグナルs1は、時刻cとdの間にp2が受信している。時刻bとcの

*A debugging mechanism for CHILL self environment
†Hideki SASAKI, Norio SATOH
‡NTT Software Laboratories

間にプロセス p2 が送信したシグナル s2 は、処理の流れを考えれば時刻 e で受信されている。しかし、時刻 b から e までは制御は p2 にあるので、p1 の仮想時間では時刻 b と e は同一である。よって s2 は時刻 b で受信されたと考えることにした。これにより時刻 c において送信後受信されていないシグナルは s1 のみとなり、時刻 d においてはすべてのシグナルの送受信が完了していると見做すことができる。これは CRTL 内でのシグナル送受信の実現法とデバッガのシグナルの取り扱いが同一となることを意味する。

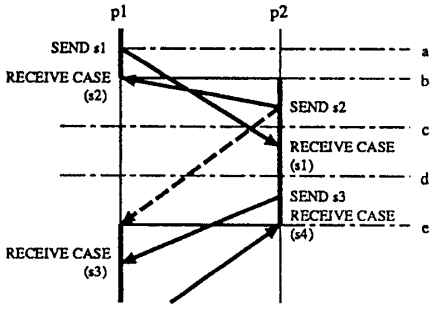

図 2: プロセス間のシグナル送受信の例

4 CHILL セルフデバッガの実現法

4.1 シグナルに着目したデバッグ

シグナルは、シグナル識別子、送信プロセス、受信プロセス、シグナルに乗せて相手のプロセスに渡す引数 の四情報を内部に持っている。シグナル送受信のデバッグをサポートするデバッガは、この情報をユーザーに提供することになる。実行停止時および実行時の各々について、実現する機能を示す。

BreakPoint で停止した状態では、CHILL プログラム内に存在するシグナルは、送信後未受信なものだけである。これと合わせて、各受信待ちプロセスの受信待ちシグナルについて、要求に従って情報を提供する必要がある。

また、BreakPoint の設定については、通常のデバッガのような関数名や行番号等の静的情報に対する設定だけでなく、プロセス間で送受信されるシグナル自体に停止トリガーを設定することを考えた。シグナル送信時に呼び出される関数内に、条件付きで bkpt 命令を埋め込むことで実現が可能である。ユーザーはデバッガに対して、シグナルネームを指定すること想定している。

実行時には、各シグナルの送受信履歴を記録しておき、ユーザーからの要求に応じて履歴を表示する。また、その履歴をもとにウィンドウ上にグラフィカルな通信履歴表示も行なう。

4.2 CRTL への機能追加

4.1で述べた機能を実現するための具体的方策について述べる。CRTL 内に保持している情報は、各プロセスのコンテクストのみである。本稿で提案するような実行時ライブラリ内にデバッグ機能を持つ構成とした場合、シンボル情報を表示するためには、事前にシンボル情報を CRTL 内に読み込む必要がある。

ユーザーが、CRTL 内のデバッグ機構を使用しようとすると、CRTL 内にシンボル情報のテーブルが作成されていない場合は、実行可能ファイルの symbol table と string table を読み込み、内部のテーブルに情報を保持することとした。4.1で述べた機能を実現するために必要なシンボル情報は、プロセスネームとシグナルネームである。CRTL 内にはプロセスの実行開始アドレス (プロセスネームと一意に対応する) とシグナル識別子 (シグナルネームと一意に対応する) を持っている。よって実行可能ファイル内のテーブルを読み込んで内部に保持しておくことで、シンボル情報の表示が可能となった。

プロトタイプとして、プロセスごとの受信待ちシグナルの表示、プロセスごとの受信シグナルキュー内のシグナルの表示、シグナル送信履歴の表示機能を CRTL に追加した。実行中の画面の一例を図 3に示す。

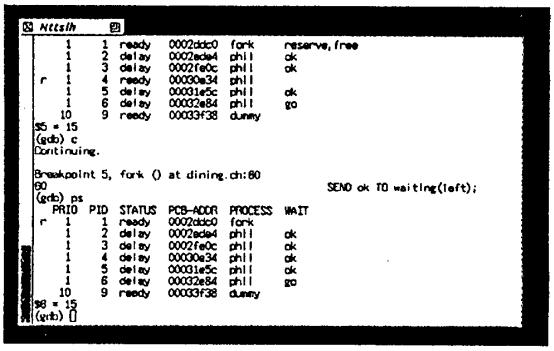

図 3: gdb を用いた CHILL デバッガ

5 おわりに

実行時ライブラリへの機能追加と関数コール機能を持つ C 言語用デバッガの組合せで、CHILL セルフ処理系における実用的なデバッガが簡単に実現できることを示した。

今後は、並列プロセスにおけるシグナル以外の同期・通信機能に対するデバッグのサポート、通信状態のウィンドウ表示、CHILL と C のシンタックスの相違を吸収するフィルタについて検討する予定である。

参考文献

[1] "CCITT HIGH LEVEL LANGUAGE(CHILL)", CCITT Recommendation Z.200, 1988

[2] 佐藤, 大森 :「マイクロコンピュータ用ソフトウェア開発に向けた CHILL 言語処理システムの実現」, マイクロコンピュータとワークステーション, 53-1, 1988.12

[3] N. Satoh, K. Ohmori : "CONSTRUCTION OF A CHILL ENVIRONMENT USING GENERALLY AVAILABLE TOOLS", Proceedings of the 5th CHILL Conference, March, 1990

[4] 大森, 佐藤 :「マイクロプロセッサ用 CHILL 言語処理におけるデバッグ支援環境」, 信学会全国大会, 3-19, 1989