リアルタイムシステム用シンボリックデバッガ (rsdb)

1R−2

雑賀 敏昌* 保里 裕之* 川俣 達也* 石川 隆* 佐藤 秀樹**

*日立プロセスコンピュータエンジニアリング(株), **(株) 日立製作所 大みか工場

1 はじめに

我々は、リアルタイム処理を指向したオペレーティングシステムの設計開発を進めている。

UNIX[1]において、プログラムデバッグ支援の為のツールとしては、sdbが広く使われるようになっている。しかし、複数のプログラムにて共用するプログラムのシンボリックデバッグを行おうとすると、効率の良いプログラム開発の為に、より強力なデバッガが必要である。

この為、複数のシンボルテーブルを参照し、共用プログラムをデバッグすることを可能にしたリアルタイムシンボリックデバッガ(rsdb)を開発した。

本稿では、rsdbにおける共用プログラムのデバッグの特徴及び実現法について述べる。

2 概要

一般にオンラインリアルタイム処理システムでは、性能条件が厳しいため、処理能力やメモリを考慮して、プログラムをリエントラント化し、図1に示すようにタスク間で共通に使用する形態を採用している。

各タスクをデバッグする場合において、参照するそれぞれの共用サブプログラムも含めて行おうとすると、デバッグ対象タスクのシンボルテーブルに加えて各々の共用プログラムのシンボルテーブルを管理し、デバッグ情報を適宜読みこむことができるようにする必要がある。

本デバッガは全ての共用プログラムがどのタスクから参照された場合においても、デバッグ出来るように設計されている。

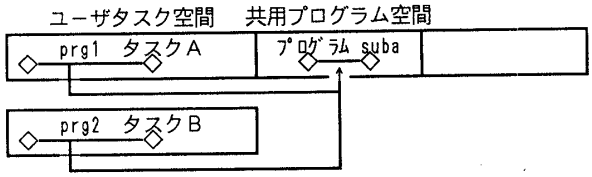


図1: タスク間共用プログラムの構成

3 rsdbの特徴

3.1 プログラム単位のシンボル情報管理

rsdbが対象とする被デバッグプログラムのシンボルテーブルは、プログラム格納エリアの容量節約の為、オブジェクトファイルから分離し、各プログラム単位に管理されている。

3.2 シンボルテーブルの動的ローディング

rsdbは被デバッグタスクと格納エリアが分離されて管理されている共用プログラムのシンボリックデバッグをサポートする為、シンボルテーブルをrsdb起動直後の初期設定時だけでなく、シンボリックアクセス指定時にも、動的にシンボルテーブルを読み込む。

更に今回のrsdbにおいては、プログラムアクセステーブル・シンボルテーブルを複数のプログラムに対して、サポート出来るように新たに図2に示すプログラムテーブルを作成する。

rsdbが共用プログラムを参照するためにプログラムテーブルを利用する処理は図3に示す。

rsdbは、まず参照指示されたプログラムのシンボルテーブルを読み込み済みか否かをプログラムテーブルで調べる。読み込み済みならば対応するプログラムアクセステーブルを用いてカレント行、カレントファイルを設定する。シンボルテーブルを読み込み未済みならば、指定プログラムのシンボルテーブルを読み込み、プログラムテーブル登録、及びプログラムアクセステーブル作成後カレント行カレントファイルを設定する。

プログラムテーブルは、1プログラムに1エントリ作成する。

Symbolic Debugger for Real-time System
Toshimasa SAIKA, Hiroyuki HORI, Tatsuya KAWAMATA, Takashi ISHIKAWA, Hideki SATO
* Hitachi process computer engineering, Inc.
** Hitachi, LTD. Omika Works.

[1]UNIXは、AT&T社のBell研究所で開発されたOSです。

これらにより、被デバッグタスクが多数の共用プログラムを使用していても、デバッグしたい共用プログラムのシンボルテーブルだけを読み込むことで、デバッグが可能となり、シンボルテーブル格納用のメモリ使用量の節約がはかれた。

また、一度シンボルテーブルを読み込めば2度目以降の参照時にはシンボルテーブル読み込みの為のディスクアクセスせずにデバッグ出来るため、応答性の高いデバッガを実現できた。

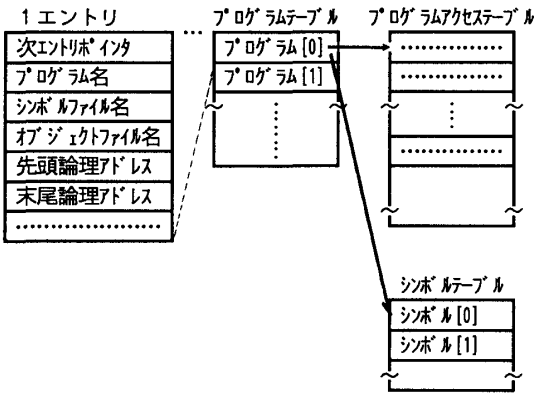


図 2: プログラムテーブル構成図

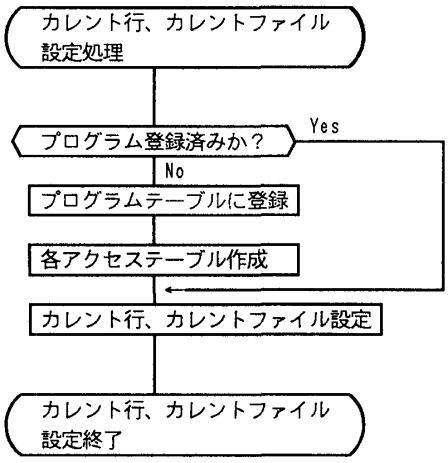
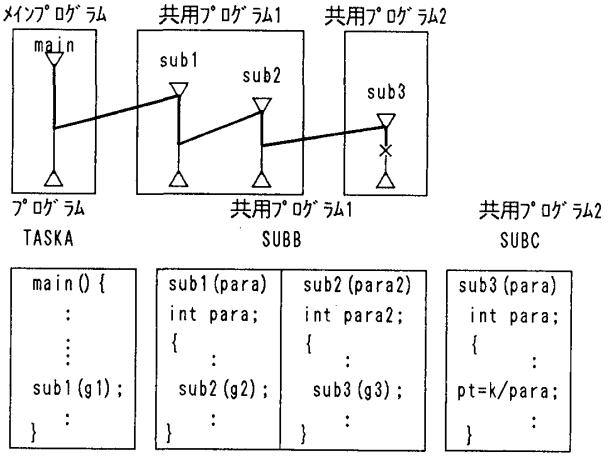


図 3: 動的ローディング処理

4 使用例

図4に複数の共用プログラムを使用するプログラムにおいて、共用プログラムでエラーが発生した時に、シンボリックにデバッグする具体的解析例を示す。

ここでは、main → sub1 → sub2 → sub3 の順にリンクし、共用プログラム sub3 内の "pt=k/para" にてエラーが発生し、その時の変数 para の値は、0 であることがわかる。




```
% rsdb main                              # rsdbの起動
0xac0ce10c in sub3:21: pt=k/para;  # 停止行の表示
*para                                   # ｶﾚﾝﾄ関数(sub3)内の変数para
0                                       # の値表示
*t                                      # スタックトレース
sub3(para=0)    [sub3:subc.c:21]    # 関数sub3が引数para=0でCALL
sub2(para2=9)   [sub2:subb.c:10]    # 関数sub2が引数para2=9でCALL
sub1(para=5)    [sub1:subb.c:19]    # 関数sub1が引数para=5でCALL
main(1,2147466896,2147466904)   [main:main.c:14]
*e
sub3() in "subc.c" (sub3)
*q
```

図 4: 使用例

5 おわりに

本稿では、共用プログラムのシンボリックデバッグの手法として、複数のシンボルテーブルサポートを紹介した。

今後は、シンボリックデバッガとの統合や、新しいアーキテクチャへの活用を考えていきたい。

参考文献

[1] AT&T, *"UNIX Sysytem V Release 3 Programmer's Guide Issue 1"*, AT&T, chap. 15, (1986)

[2] 野木、中所, "プログラミングツール", 昭晃堂, p.p 179–181, (1989)