3N-8

組込み向けOSにおける障害解析性向上の一手法

摂津　敦[†]、茨木　康之[‡]、菅井　尚人[†]、山口　義一[†]

[†]三菱電機（株）情報技術総合研究所

[‡]三菱電機（株）電力・産業システム事業所

1．はじめに

OS の障害解析手法の１つに、クラッシュダンプ機能がある。クラッシュダンプ機能は、障害発生時の主記憶内容を HDD に保存し、再起動後にその内容を解析することにより、詳細な障害原因を追求することができる。しかしながら、屋外に設置されるような組込み制御機器では、HDD などの大容量の二次記憶を持たないためクラッシュダンプ機能をそのまま適用することができない。

本稿では、クラッシュダンプ機能を、組込み制御機器に使用する組込み向け OS に適用するための課題および実現方法について述べる。

2．クラッシュダンプ機能とは

クラッシュダンプ機能とは、障害発生時に主記憶内容をそのままダンプデバイス（基本的にはスワップデバイス）に吐き出した後に H/W リセットを行ってシステムを再初期化し、OS 再起動後にユーティリティを用いてデバイスにアクセスすることにより、障害発生時の OS/アプリケーションの状態を解析できるようにするものである(図 1)。この機能は、Solaris、HP-UX、FreeBSD, Linux(参考文献[2]),WindowsNT などの汎用 OS で採用されている。

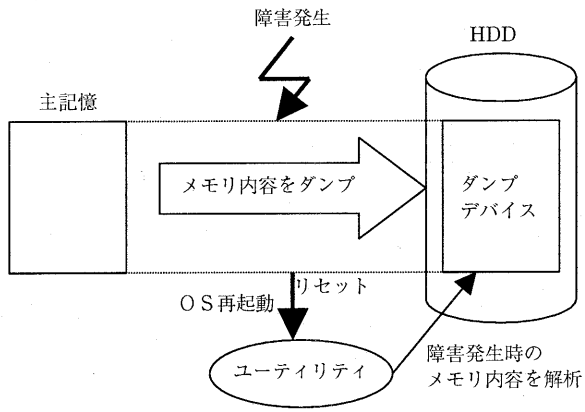

図 1：クラッシュダンプ機能

クラッシュダンプ機能では、H/W として実装されている主記憶内容を全て保存するため、障害発生時に動作していたプロセスの状態など、障害発生時におけるシステム状態および障害発生に至るまでのシステム状態を詳細に解析することが可能である。

3．組込み用途での課題

組込み制御機器では、耐環境性の側面から機械部分が存在する HDD は使われず、バッテリバックアップ付きメモリやフラッシュ EEPROM などの電子デバイスが二次記憶として用いられる。よって、クラッシュダンプにおけるダンプ先デバイスにはこれら電子デバイスを選択する必要がある。

これら電子デバイスは、HDD に比べ高価であり、容量も限られるため、主記憶の容量よりも少ない場合が多い。また今後、電子デバイスの容量が増えても、それに連動して実装される主記憶容量も増えることが予想される。従って、このような組込み制御機器では、メモリを全てダンプデバイスに書き込むクラッシュダンプ機能をそのまま適用することはできず、ダンプする主記憶容量を削減することを検討する必要がある。

4．解決方法

ダンプする主記憶容量を削減するためには、障害解析に必要なデータを選別し、その領域のみダンプデバイスに書き込む方法が考えられる。以下に、障害解析に必要なデータについて考察する。

4．1　ダンプ対象共通データの選別

基本的にクラッシュダンプ機能が必要となる障害はドライバを含めたカーネル内で発生する。ユーザプログラム（プロセス）で発生した障害は、ＯＳ内で処理できる（コアを吐く、シグナルを送信する）からである。このため、実際の障害解析に最低限必要なデータは以下のものであると考える。

1) 障害発生時の CPU レジスタ情報
　障害発生時の CPU の動作状態を把握するため。
2) ページング用変換テーブル(ページテーブル)
　通常、機器は仮想記憶で動作しているが、ダン

An enhancement for trouble analysis in embedded OS.

Atsushi SETTSU, Yasuyuki IBARAKI,
Naoto SUGAI, Yoshikazu YAMAGUCHI
Mitsubishi Electric Corp.

デバイス出力部へデータを渡すタイミングは，バッファが満たされた場合である．鶴から受け取ったデータがバッファの残りのサイズより大きい場合は，バッファを満たせるところまで満たしその内容をデバイス出力部へ渡した後に，残りのデータをバッファに蓄積する．これによって，鶴がトレースデータを出力する場合にデバイス特有のレコードサイズを意識する必要がない．

2.3 デバイス出力部

2.3.1 デバイス初期化ルーチン

デバイスの初期化処理を行う．また，初期設定部からログデバイスの I/O アドレスを受け取り，その値をデバイス出力部の大域変数として保持する．

2.3.2 データ書き出しルーチン

データをデバイスへ書き出す処理を行う．鶴が処理を行っているときは，割込みを使用できないため，書き出し時のオーバヘッドを減らす工夫が必要である．本ルーチンでは，データのコピーによるオーバヘッドを抑えることと，デバイス特有のレコードサイズで書出し可能とするために，バッファ蓄積部のバッファへのポインタとそのサイズを受け取る．

本論文では特に，HDD と Ethernet を用いたデータ書き出しルーチンを開発した．以下，それぞれについて述べる．

2.3.3 HDD

IDE-HDD への最小書き出しサイズは 512byte であり，実際の書出しサイズはこの倍数となる．書出しサイズをバッファ蓄積部のバッファサイズに等しくすることで，書出し動作の回数が 1 回になり，オーバヘッドを減らすことができる．そして，さらにオーバヘッドを減らすために，データは特定のファイルシステムに変換することなく，バッファ蓄積部のバッファの内容をそのままで HDD にバイト列として書出す．また，複数のデータを保存できるように，データの最後に終了レコードセクタが付加される．記録されたデータを鶴 Tools が解析可能とするために，データをファイルに書き出すプログラムは別途作成している．

2.3.4 Ethernet

Ethernet の送信バッファは 1.5Kbyte である．トレースデータは，信頼性が保証されなければならない．しかし同時に，出力時のオーバヘッドも減らさなければならない．Ethernet 出力を行う際には，一般的なプロトコルとして TCP と UDP が挙げられる．TCP はデータの信頼性は保証されるが，オーバヘッドが大きい．一方，UDP はデータの信頼性は保証されないが，オーバヘッドが少ない．したがって，ここでは UDP に簡素なデータ保証機構を付加したプロトコルを実装し使用する．このデータ保証機構は，Ethernet デバイス出力部と受信プログラムでパケットの到達を確認し，パケットが失われた場合は，パケットの再送を行う．

2.4 インターフェース

本ログ機構で提供する機能を利用するために実装したインターフェースは以下の 2 つである．

- int trace_dev_open(char *ptr)

  ユーザから，カーネルに認識されたログ対象デバイスの名前を受け取る．これにより，ユーザの本ログ機構の利用が可能になる．

- int store_data(unsigned char*ptr, unsigned int *size)

  デバイスへ出力するデータを受け取る．

3 おわりに

本論文では，鶴のログ機構の設計と実装について述べた．本ログ機構によって，鶴のトレースデータをデバイスへ出力することが可能となった．また，デバイス依存，非依存の処理を分けたことにより，ログ出力対象デバイスの追加もできることを示した．また，本ログ機構は鶴以外でも適用が可能である．今後は，本ログ機構のオーバヘッドを測定し，バッファ蓄積部のバッファサイズや，データ書き出しルーチンの最適化を図る予定である．

参考文献

[1] 森本，池谷，毛利，吉澤：OS の性能評価を可能とする命令トレーサの開発，情報処理学会研究報告 2000-OS-84，pp.157-164，2000.

[2] 森本：命令トレーサによる性能評価システムの開発，東京農工大学卒業論文，1999.