4ZA-1

インテル x86 系 CPU 上で動作する組み込み用 OS 「開聞」の設計と試作

金原　学†　　並木　美太郎††
† 東京農工大学工学部　†† 東京農工大学大学院工学教育部

1　はじめに

組み込みシステムでは，サーバなどのマルチユーザ的なアプリケーションや，カーナビなどのシステムなど，システムの高機能化にともない，メモリプロテクションが必要になっている．このような，組み込みシステムの複雑化，大規模化に伴い，CPU にも MMU を持つものが増加している．しかし、ITRON や eCos などの既存の組み込み OS は，ローエンドの CPU を考慮して設計されているため，メモリ管理で保護機能を持っていない [1][2]．近年では，インテル x86 系 CPU のようにセグメント機構やページングを持った CPU を搭載したシステムも増加しているが，この種の組み込み用 OS では MMU を有効できていない．組み込み Linux では，ページングを行うが資源の少ない組み込みシステムで，組み込み Linux を適用できない場合もあるので，小型軽量の組み込み用 OS の意義は現在も高い [3].

そこで本稿では，インテル x86 系 CPU のセグメントおよびページング機構を用いたメモリ管理を示し，筆者の所属する研究室で開発している組み込み用 OS「開聞」を x86 系 CPU に移植した結果について述べる．

2　目標

組み込み用 OS で稼動する各タスクの機械語およびデータをタスク間で保護すると同時に，共有も可能とする．一般に組み込みシステムでは，主記憶上に各種プログラムおよびデータが展開されることが多いので，タスクの実行に必要な機械語、データおよびスタック以外にも応用プログラムで必要となる各種データも同様の機構で保護・共有が可能になることを目指す．これらの保護・共有は，インテル x86 系 CPU 上のセグメント機構を用い，タスクのアドレス空間をセグメントにより保護および共有を行う．

3　「開聞」のメモリ管理機構

3.1　機能と概要

「開聞」では，タスクが使うメモリ領域をメモリオブジェクトを用いて管理している．メモリオブジェクトは，タスクの使うメモリ領域内の連続した領域を指し，またその他の情報を持つ構造体である．図 1 にメモリオブジェクトとタスクのアドレス空間との関係の概念図を示す.

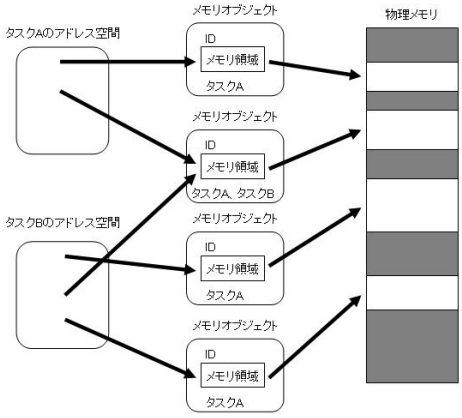

図 1 メモリオブジェクトとタスクのアドレス空間

図 1 のように，タスクのアドレス空間は複数のメモリオブジェクトによって形成されている．通常タスクは，自身のアドレス空間にコード，データ，スタック領域を持つ．これらの領域も 3 つのメモリオブジェクトとして管理される．しかし,「開聞」ではタスクの持つメモリオブジェクトを 3 つに制限していない．つまりタスクはこの他にも，複数の連続したメモリ領域を，アクセスすることが可能であり，さらにそれらのメモリ領域を他のタスクと共有することもできる．また，メモリオブジェクトは OS により大きさ，属性，参照可能なタスクなどが管理されている．これによりタスク間でアドレス空間の，メモリオブジェクト単位の保護と共有が実現できる．

3.2　システムコール

タスクは，システムコールを用いてメモリオブジェクトを生成したり，生成したメモリオブジェクトが指すメモリ領域にアクセスできるようになる．次の表 1 にメモリオブジェクトに関する主なシステムコールを

Design and Prototype of the Embedded Operating System ”Kaimon” for Intel x86 Processor

† Manabu Kinbara
Faculty of Engineering, Tokyo University of Agriculture and Technology
†† Mitaro Namiki
Graduate School of Technology, Tokyo University of Agriculture and Technology

示す.

表1 メモリオブジェクト関連のシステムコール

名称	パラメータ	処理
CreateMobj(char* mobjID int base; Int size; char attrribute)	メモリオブジェクトID, メモリオブジェクトが指す物理領域の先頭アドレス, メモリオブジェクトが指す, 物理メモリ領域の大きさ メモリオブジェクトの属性	メモリオブジェクトを生成する
DeleteMobj(mobjID)	メモリオブジェクトID	メモリオブジェクトを消滅する
GetMobj(mobjID)	メモリオブジェクトID	メモリオブジェクトを自身のアドレス空間に加える
ReturnMobj(mobjID)	メモリオブジェクトID	メモリオブジェクトを自身のアドレス空間から除く

次に，システムコールの使用例を示す．なお，タスク A は，タスク B よりも先に実行されるものとする．

タスク A 内の処理

```
CreateMobj(mobjID,baddr,size,attrbt);
GetMobj(mobjID);
//獲得したメモリ領域に対するアクセス
ReturnMobj(mobjID);
```

タスク B 内の処理

```
GetMobj(mobjID);
//獲得したメモリ領域に対するアクセス
ReturnMobj(mobjID);
DeleteMobj(mobjID);
```

この例では，タスク A 内でメモリオブジェクトを生成し，タスク B が消滅させている．その間に，タスク A もタスク B も同一のメモリオブジェクト mobjID にアクセスしている．

4 内部設計

4.1 メモリオブジェクトの管理

メモリオブジェクトの実体は構造体であり，表 2 のようなメンバを持つ．OS は，メモリオブジェクトを双方向リストでつなげて管理している．これらの情報から，OS はメモリオブジェクトと物理メモリ領域の対応，属性，および参照可能なタスクを管理している．

4.2 メモリオブジェクトとセグメント

インテル x86 系 CPU では，セグメント機構が用意されている．セグメントは，メモリ領域を定義するメカニズムであり，タスクごとに複数定義することもできる．またセグメントは大きさや属性の設定が可能で，これらに違反するメモリアクセスを行うと，CPU は例外を発生し OS に処理を渡す．OS は，メモリオブジェクトの情報に沿ったセグメントを内部で定義することで，セグメントの保護機能を利用し，メモリオブジェクトを管理している．

表2 メモリオブジェクトの内部構造

メンバ	説明
typeList *list	メモリオブジェクトの双方向リスト
char *mobjID	メモリオブジェクトを識別するための名前
int baddr	メモリオブジェクトが指物理メモリ領域の先頭アドレス
int size	メモリオブジェクトが指す物理メモリ領域のバイト単位の大きさ
char attribute	メモリオブジェクトへのアクセス可能な形式を表す属性, read,write,execute の3種類がある
int tid	メモリオブジェクトを参照しているタスクのID
int *next int *pre	メモリオブジェクトを参照しているタスクIDの双方向リスト

5 試作と評価

3.2 で示したシステムコールの実行クロック数を計測したところ，次のような結果になった．なお，計測時に用いたプロセッサは pentium　800Mhz である．また，表中の“ 獲得したメモリオブジェクトへのアクセス ”は，異なるセグメントへアクセスするための時間を計測している．

システムコール	実行サイクル数
CreateMobj	847
DelteMobj	506
GetMobj	624
ReturnMobj	487
獲得したメモリオブジェクトへのアクセス	75

6 おわりに

メモリオブジェクト単位でのメモリ管理を提案し，インテル x86 系 CPU のセグメント機構を用いて実現することができた．しかし，現段階では「開聞」はページングを実装していない．今後の課題として，本稿で提案したメモリ管理法に，どのようにページングの機能を追加するか検討が必要である．

参考文献

[1] (社) トロン協会 ITRON 仕様検討グループ **μITRON4.0 仕様 ver.4.02.00** *2004* TRON ASSOCIATION,JAPAN

[2] **eCos リファレンスマニュアル** *1998* Cygnus Solutions

[3] **組み込み Linux 入門** *2003* CQ 出版株式会社