2V-3

unix*のリアルタイム性向上へのアプローチ[1]

斎藤 正史　山口 義一　伊藤 均　水野 忠則　渡辺 治

三菱電機(株)情報電子研究所

1．まえがき

unixはｴﾝｼﾞﾆｱﾘﾝｸﾞﾜｰｸｽﾃｰｼｮﾝ用のOSとして産業標準の地位を得つつある。しかし、unixは元々TSS向きに開発された為に、ﾜｰｸｽﾃｰｼｮﾝで要求されるﾏﾝﾏｼﾝｲﾝﾀﾗｸｼｮﾝに対する即応性に問題がある。そこで、使用者のﾜｰｸｽﾃｰｼｮﾝへの要求指示(例えばﾏｳｽのｸﾘｯｸによって)に対して、一定の短い時間でｻｰﾋﾞｽが行なえるようにｱﾌﾟﾘｹｰｼｮﾝを構築できるように、unixを改造してﾘｱﾙﾀｲﾑ性を強化する手法について考察した。

2．ﾘｱﾙﾀｲﾑ性の要求

ﾜｰｸｽﾃｰｼｮﾝはﾏﾙﾁｳｨﾝﾄﾞｳ機能やﾘﾓｰﾄﾌｧｲﾙｱｸｾｽ機能を提供し操作性の向上を図っている。ﾘﾓｰﾄﾌｧｲﾙｱｸｾｽには通常ｸﾗｲｱﾝﾄ/ｻｰﾊﾞﾓﾃﾞﾙを適用し、要求毎に1つのﾌﾟﾛｾｽでその要求を遂行することが多い。又、ﾏﾙﾁｳｨﾝﾄﾞｳ機能もｳｨﾝﾄﾞｳを管理するﾌﾟﾛｾｽをｻｰﾊﾞ、ﾕｰｻﾞをｸﾗｲｱﾝﾄと見立てたｸﾗｲｱﾝﾄ/ｻｰﾊﾞﾓﾃﾞﾙのように構築するとﾓｼﾞｭﾗﾘﾃｨが高くなり望ましいｿﾌﾄｳｪｱ構成となる。しかしその反面、通常ｱﾌﾟﾘｹｰｼｮﾝの他にﾌﾟﾛｾｽ数が増加しｽｹｼﾞｭｰﾙが頻繁に起こる。

ﾘﾓｰﾄﾌｧｲﾙｱｸｾｽやｳｨﾝﾄﾞｳ管理のﾌﾟﾛｾｽはﾕｰｻﾞのｲﾝﾀﾗｸｼｮﾝを考えると処理の即応性が要求される。従って、これら特定ﾌﾟﾛｾｽに対する高速起動が必要となる。unixｶｰﾈﾙはﾃﾞﾊﾞｲｽ等の入力待ちでsleepしているﾌﾟﾛｾｽに完了割り込みが上がるとそのﾌﾟﾛｾｽをwakeupする。しかし、このﾌﾟﾛｾｽはready状態になるだけであり、すぐにｽｹｼﾞｭｰﾘﾝｸﾞされない可能性もある。これでは応答の均一性が低いｼｽﾃﾑになりやすい。従って、特定のﾌﾟﾛｾｽを高速にｽｹｼﾞｭｰﾘﾝｸﾞするためにはunix本来のﾌﾟﾛｾｽ管理の形態では難しく、改善が望まれている。

*: Unix OS was developed by Bell Laboratories and is licenced by AT&T.

3．ﾌﾟﾛｾｽの高速起動

3．1　設計指針

ﾌﾟﾛｾｽの高速起動を実現するにあたり、以下の設計指針を立てた。

①unixｶｰﾈﾙの変更は極力抑える。

②ｲﾍﾞﾝﾄが数msec単位に起きるような事象の制御も可能とする。

③ｼｽﾃﾑに固有のﾌﾟﾛｾｽしか扱えないのではなく、通常ｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｾｽから自由に移行可能とする。

④ｲﾍﾞﾝﾄにより起動されるようなものだけではなく、通常ｱﾌﾟﾘｹｰｼｮﾝﾌﾟﾛｾｽからも起動可能とする。

3．2　方式検討

設計指針を基に、以下の3方式を考案した。

①ｼｽﾃﾑﾌﾟﾛｾｽ(SP)方式

unixのｽﾜｯﾊﾟｰで従来より行なわれている、unixｶｰﾈﾙとﾌﾟﾛｸﾞﾗﾑﾃｷｽﾄを共有した形で実現する(図1)。ｲﾍﾞﾝﾄにより起動されるﾘｱﾙﾀｲﾑﾌﾟﾛｾｽはunixｶｰﾈﾙのﾃﾞﾊﾞｲｽﾄﾞﾗｲﾊﾞのように組み込み、そうでないものはunixのﾌﾟﾛｾｽ間通信機能を利用して起動されるように構築する。ﾘｱﾙﾀｲﾑﾌﾟﾛｾｽはｼｽﾃﾑの立ち上げ時に生成され、停止時まで存在する。unixｶｰﾈﾙの種々の機能を利用可能であるが、H/W割り込みのマスク等の処理に充分注意を払う必要がある。

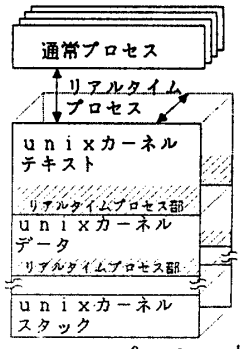

図1．システムプロセス方式

An Approach for Realtime Enhancement to Unix OS [1]

Masashi SAITO, Yoshikazu YAMAGUCHI, Hitoshi ITO,
Tadanori MIZUNO, Osamu WATANABE
Mitsubishi Electric Corp.

②ｼｪｱﾄﾞﾗｲﾌﾞﾗﾘ(SL)方式

unixｶｰﾈﾙは通常ﾌﾟﾛｾｽが最初に起動(fork/exec)される時にﾌﾟﾛｾｽの論理空間の固定領域にunixのｼｪｱﾄﾞﾒﾓﾘ機能を利用してﾘｱﾙﾀｲﾑﾌﾟﾛｾｽのﾌﾟﾛｸﾞﾗﾑｺｰﾄﾞを割り付ける（図２）。すべての通常ﾌﾟﾛｾｽに対してこの割り付けを行なう為、どの通常ﾌﾟﾛｾｽが実行中でも論理空間の切り替えが不必要となり、要求があるとすぐに走行可能となる。

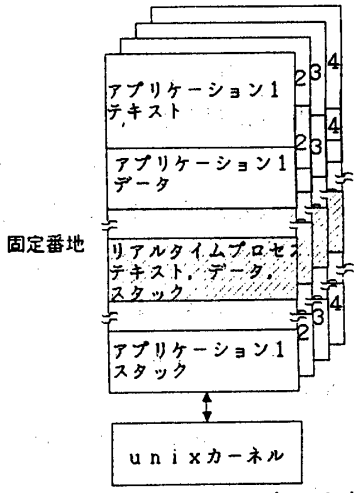

図２．シェアドライブラリ方式

③ｼｽﾃﾑの非ﾌﾟﾛｾｽ(NSP)方式

unixｶｰﾈﾙは通常ﾌﾟﾛｾｽからのｼｽﾃﾑｺｰﾙにより、ﾘｱﾙﾀｲﾑﾌﾟﾛｾｽ用のﾒﾓﾘ領域を動的に確保しそのｺｰﾄﾞを割り付ける（図３）。ﾘｱﾙﾀｲﾑﾌﾟﾛｾｽは通常ﾌﾟﾛｾｽのようにそれぞれ論理空間を持つのではなく、unixｶｰﾈﾙ空間の一部を使用して実行される。又、ﾌﾟﾛｾｽ管理情報として専用の小さな制御ブロックのみを保持している。従って、高速起動は可能であるが、通常ﾌﾟﾛｾｽとは別管理が必要となる。

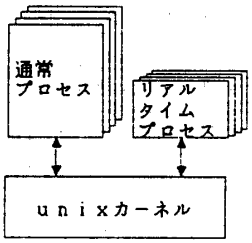

図３．システム非プロセス方式

３．３　評価

SP方式は組み込みｼｽﾃﾑとしては充分であるが、ﾘｱﾙﾀｲﾑﾌﾟﾛｾｽの登録にOSの再生成が必要でありｼｽﾃﾑ管理者の負荷を増大させる。又、ﾃﾞｨｽｸ容量もOS作成用に大量に必要となり、個人で使用するﾜｰｸｽﾃｰｼｮﾝ用としては現実的でない。

SL方式は通常ﾌﾟﾛｾｽの暴走によりﾘｱﾙﾀｲﾑﾌﾟﾛｾｽを破壊する可能性があり好ましくない。又、ﾘｱﾙﾀｲﾑﾌﾟﾛｾｽのために通常ﾌﾟﾛｾｽの起動(exec)にｵｰﾊﾞﾍｯﾄﾞがかかるのは好ましくない。

NSP方式は2種のﾌﾟﾛｾｽの管理が必要であるが、通常ﾌﾟﾛｾｽとの独立性も高く良い方式である。又、性能面でも十分に満足するものと考える。

以上３方式を性能・OS／通常プロセスとの親和性の点で検討を加えた結果を表１に示す。

表１．各方式の評価

評価事項　　方式	SP	SL	NSP
性　　能	×	△	○
ｲﾝﾌﾟﾘﾒﾝﾄの容易性	×	△	○
OSとの親和性	△	○	△
通常ﾌﾟﾛｾｽとの親和性	○	×	○

○：良い　　△：普通　　×：悪い

比較検討の結果、総合的にはNSP方式が優れていることがわかった。

４．あとがき

現在、NSP方式のｲﾝﾌﾟﾘﾒﾝﾃｰｼｮﾝにおけるunixｶｰﾈﾙへの追加ｼｽﾃﾑｺｰﾙの設計を行なっている。追加するｼｽﾃﾑｺｰﾙはﾘｱﾙﾀｲﾑﾌﾟﾛｾｽの生成／消滅や起動／停止等よりなる。

この拡張機能設計終了後、ｲﾝﾌﾟﾘﾒﾝﾃｰｼｮﾝと性能評価を行ないﾜｰｸｽﾃｰｼｮﾝ用として効果的なﾘｱﾙﾀｲﾑ性能の良いOSとする予定である。

参考文献

[1]山口，他：unixのﾘｱﾙﾀｲﾑ性向上のｱﾌﾟﾛｰﾁ [2]，
情報処理学会第33回全国大会，1986