マイクロコンピュータと
ワークステーション 54－4
（１９８９．２．２１）

ＵＮＩＸにおけるリアルタイム性の導入に関する一考察

市瀬規善，水橋由紀子，古城隆，（日本電気マイコンテクノロジー（株）），
永作浩之（日本電気アイシイマイコンシステム（株）），
門田浩（日本電気（株））

ＵＮＩＸは元々ＴＳＳ用に開発されたＯＳであるため，即応性に問題がありリアルタイム制御との整合性が悪いといわれてきた．今回，日本電気では，この様なのＵＮＩＸの問題点を解決し，３２ビットオリジナルマイクロプロセッサＶ６０／７０上のリアルタイムＵＮＩＸであるＲＸ－ＵＸ８３２を開発した．これは，システムの大規模化，多様化，多機能化に対応して，リアルタイム制御を実行しながらマンマシンインタフェースやデータ処理などのようなＴＳＳ処理も両立するためのＯＳである．
本編では，ＵＮＩＸにおけるリアルタイム性の導入の問題点とその解決法を，ＲＸ－ＵＸ８３２を通して述べていく．

SIGMIC 54-4
An issue in introduction of real-time capabilities into UNIX

Noriyosi Ichinose, Yukiko Mizuhasi, Takashi Kojo(NEC Microcomputer Technology Ltd., 2-13-11 Shibaura, Minato-ku, Tokyo 108, Japan), Hiroyuki Nagasaku(NEC IC Microcomputer Systems Ltd.), and Hiroshi Monden(NEC Corporation)

UNIX operating system, which have been developed originally for Time Sharing System (TSS), is said to lack of real-time capabilities. We, in NEC Corporation, have overcome such problems, and have developed RX-UX832 which is a realtime UNIX on NEC's 32-bit original micro-processors V60/70. This is the operating system manages TSS and realtime operating simultaneously.
In this paper, we discuss problems and solutions in introduction of real-time capabilities into UNIX operating system.

1．はじめに

マイクロプロセッサが高性能，高機能になるにつれ，それが組み込まれるシステムの性能，機能も飛躍的に向上している．これに伴い，従来，大型かつ複雑なシステムで要求されてきた用件がにわかに必要となっている．たとえば，Factory Automation システムでは工場内の高価な機械をコントロールするために，耐故障性が要求されたり，運用時にシステム構成を切り換え（例：装置の増設，入れ替えなど）たりする必要がでている。また，外部ファイルに大量のデータを蓄えたり、運用中に生じる種々の事態に対してのマンマシンインタフェースも必要になっている．さらに，システムの機能の増加によりそのソフトウエア開発量の増大およびその開発効率が問題となってきている．

一方，ソフトウェア開発環境としては，ＵＮＩＸベースのワークステーションが標準になってきているが，近年，従来のＴＳＳ処理だけでなく，リアルタイム処理も行えることが望まれている．研究室や実験室などでの Laboratory Automation では，ＵＮＩＸの環境を利用しながら，同時にリアルタイムに測定機器を制御したり，ネットワーク処理を行えることなどが必要になってきている．

上記のような要望に応えるため，マイクロプロセッサ組み込み型のリアルタイムＯＳの機能と，インタフェースや操作性および汎用的な機能を豊富に持つＵＮＩＸの様なＴＳＳ型の機能を合わせもつＯＳが必要となってきており，今回，日本電気では，３２ビットオリジナルマイクロプロセッサＶ６０／７０上のリアルタイムＵＮＩＸであるＲＸ－ＵＸ８３２を開発した．

本編では，ＵＮＩＸとリアルタイムＯＳという特性の異なる二つのＯＳの統合における問題点とその解決法について，ＲＸ－ＵＸ８３２を通して述べていく．

2．リアルタイムＯＳとは

リアルタイムシステムの定義は次のように宣言できる．

> ”リアルタイムシステムとは，ある事象が発生した後，ある一定時間内に所定の処理を終了するシステムである．”（図2．1）

例えば，走行ロボットなどの制御システムはリアルタイムシステムであるべきであるが，これの障害物回避行動を例にとると，まず，このシステムは障害物を発見した場合，障害物との衝突を避け，適切な回避行動をとらなければならない．この場合に，障害物の発見が”事象の発生”であり，障害物との衝突までの時間が”一定時間”であり，適切な回避行動が”所定の処理”に当たる．

このような定義の元で，リアルタイムＯＳに要求されるのは次のことである．

> ”リアルタイムＯＳは，リアルタイムシステムをユーザが構築できるようにサポートしなければならない．つまり，ユーザが事象が起こってから所定の処理が終了するまでにかかる最悪時間を予測できなければならない．”

このために，ＯＳに要求される機能は次の通りである．

1） 応答時間の最悪値が確定していること
　－ある事象が発生したのち，ある一定時間内にユーザの指定する処理を開始できること．またその時間が短いこと．この時間をリアルタイム応答時間と呼ぶ

これによってユーザは，指定する処理が許される最大時間を確定でき，しかも，この応答時間が短い程，最大時間が長くなる．

しかし，ここで重要なのは，高速性ではなく確定性であることに注意が必要である．

2） ユーザが指定する処理が必要とする最悪時間が確定すること

（1）カーネル内走行に要する時間の最悪値が確定できること

システムコールなどの走行時間の最悪値が確定できることをさす．特に，その中で待ち状態にはいる可能性のあるものは，問題である．

（2）上記以外の部分の走行に要する時間の最悪値が確定できること

ＵＮＩＸなどで採用されているデマンドロードなどは走行に要する時間の確定性を失わせるので問題である．

3） ノンストップ性
　－システムが常に作動していること．または，故障などで止まる確率が低いこと．

システムが，障害によって止まってしまってはリアルタイム性は保証できないという点で重要である

これらをもってリアルタイムＯＳと呼ぶ．

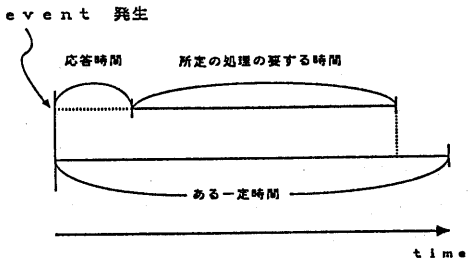


図2．1　リアルタイム処理

3．ＵＮＩＸのリアルタイム性能上の問題点

リアルタイムシステムは，その性格上，前節で述べたような性能および機能面での特徴が存在する．ＵＮＩＸをリアルタイムシステムのＯＳとしてそのまま使用すると，次のような点が問題となる．

1)タスクのスケジューリング

ＵＮＩＸはもともとタイムシェアリングシステム用のＯＳであるため，基本的なタスクのスケジューリングは，それぞれのタスクに順番にＣＰＵをある一定時間ずつ割り当てる方式を採用している．またタスクの優先度はＵＮＩＸカーネルがタスクの走行時間と連続待ち時間とから決定する．このためユーザーはタスクの実行制御の順番に関してある程度の操作は可能であるが，事象発生からタスクが動き出すまでの応答時間の予測はできない．

2) メモリの管理

ＵＮＩＸでは，タスクのテキスト部やデータ部を必要に応じて２次記憶から主記憶にロードする方式を採用している．一般的に２次記憶へのアクセス速度は，主記憶へのアクセス速度と比較して圧倒的に遅い．発生した事象を処理するタスクが動きだしても，そのタスクのテキスト部やデータ部が主記憶上に存在しないときには２次記憶から読み込まなければならなくなり，リアルタイムシステムとして期待されている応答性能を満足することはできない．

3) 待ち状態の時間

ＵＮＩＸでは，タスクがファイルのアクセスを行なったとき，システムコールの中で待ち状態になることがある．このとき，待ち時間の予測はつかないし，また待ち状態でいる最大時間の指定もできない．これはタスクの走行時間の予測ができないことを意味している．

4) ファイル破壊

ファイルのアクセス速度が遅いため，ファイルの内容をメモリ上に持つことにより，実質的なファイルのアクセス回数を減らし，あるタイミングでメモリ上の内容と実際のファイルの内容との整合をとっている．このディレイドライト方式のため，システムに障害が発生した場合，バックアップとしての機能も持っているはずのファイルが破壊されている可能性が高くなっている．

後節以降では以上のような問題点を考慮した上で，ＲＸ－ＵＸ８３２での設計アプローチについて述べる．

4．設計思想

前節で述べたように，ＵＮＩＸとリアルタイムＯＳを統合する際に，２つのＯＳの性格の違いからいくつかの問題点が考えられた．産業用組込みシステム等で使われる性格上，リアルタイムシステムでは，実時間応答性能，同時走行タスク相互の干渉の排除，フォルトトレラント機能などの組込みが求められる．このような条件により，リアルタイムＯＳにＵＮＩＸインタフェースを組み合わせるときには，いくつかの困難な問題がある．本節ではさらに，リアルタイム処理とＵＮＩＸのインタフェースをひとつの環境で提供するリアルタイムＵＮＩＸに要求される条件について考察する．

1) リアルタイム性能の維持

ＵＮＩＸの存在によって，リアルタイムタスクの実時間性能が低下しないようにする必要がある．

一般に，ＵＮＩＸカーネル走行中はタスク切り換えが禁止されているが，このカーネル走行時間がリアルタイムシステムから見ると非常に長いことが問題になる．

また，ＵＮＩＸファイルシステムのアクセスは周辺機器とのデータ転送を伴うため，実時間応答性の低下が予想される．これを小さくするため，ファイルアクセスの高速化が要求される．

2) 柔軟なＯＳ構成が可能であること

小規模な組込みシステムから，大規模システムまでの幅広い分野に適応できるように，ＯＳが機能モジュール単位に分割されていて，用途に合わせて構成できることが必要である．

3) リアルタイムとＵＮＩＸの世界が隔てられたものではないこと

リアルタイムとＵＮＩＸの世界が切り離されているのでなく，両方のシステムコールを相互に利用して同期をとったり，通信したりすることができること．

また，ＵＮＩＸのファイルを共有できれば，リアルタイム処理で収集したデータをＵＮＩＸで編集したり解析したりすることが可能になる．

4) 標準ＵＮＩＸインタフェースとの互換性

ＵＮＩＸインタフェースを採用することにより，標準的な環境で，豊富な流通ソフトウェアを利用することができるようになる．

5) 開発支援環境

アプリケーションの開発やデータ処理などをＵＮＩＸ環境で提供することによって，リアルタイムアプリケーションの，開発からテスト・デバッグまでを自身の環境で効率的に行えること．

5．実現方式の検討

ＵＮＩＸへのリアルタイム性の導入は，リアルタイム処理を行うＯＳとＴＳＳ処理を行うＵＮＩＸという，ふたつの異なった特性を持つＯＳをひとつのマシン上に実現することであるが，これには次のようないくつかの方式が考えられる．

ここでは，今まで述べてきたような問題点や要求が，それぞれの方式でどのように解決されるかについて比較検討を行った．

1）仮想マシンによるアプローチ

基本ＯＳの上に仮想マシンを設定し，そこに各ＯＳを実現する．仮想マシン間の独立性が強く，機能モジュールの着脱は容易に行える．しかし，反対にＯＳ間の通信や資源の共有にコストがかかることになる．

2）一枚岩のＯＳとしてのアプローチ

ひとつのＯＳの中に異なった特性を盛り込んでいく方法で，たとえば，ＵＮＩＸカーネルに高精度の周期プロシジャの登録やプロセスのメモリ常駐化などの機能を追加してリアルタイム処理に対応させるアプローチである．この方式は，全体的に処理やデータ構造が重くなりがちで，単純で高性能を要求するＯＳの実現を困難にするという問題がある．また，機能モジュールの着脱や，独立した再開処理を実現するのは非常に困難である．

3）下位ＯＳのアプリケーションとして上位ＯＳを実現するアプローチ

下位ＯＳの１タスクとして上位ＯＳをつくり，その中に上位ＯＳの全タスクを実現する．この方式では，上位ＯＳのタスクは下位ＯＳから見えなくなっており，下位ＯＳの提供するタスク管理機構のサービスをうけることができない．従って，ディスパッチャやメモリ管理などを別個にもつことになり，上位ＯＳが複雑になりオーバヘッドも増大する．

また，下位ＯＳからは上位ＯＳが１タスクに見えるため，上位ＯＳの中の１タスクが下位ＯＳの要因で待ち状態になった場合には，上位ＯＳ全体が待たされることになり，上位ＯＳ中でのディスパッチが困難になると予想される．

4）階層構造によるアプローチ

２つのＯＳを階層化し，下位ＯＳでリアルタイムＯＳのような基本的な機能をコンパクトかつ高速に実現し，上位ＯＳではＴＳＳインタフェースのような比較的大きい複雑な機能をサポートする．この方式では機能モジュールの分離や独立な再開処理という点で有利である．また，下位ＯＳの機能を上位ＯＳのタスクから使用できるようにすることで，ＯＳ間の通信などが可能になる．

6　ＲＸ－ＵＸ８３２の実現

ここでは，ＲＸ－ＵＸ８３２の説明を通してＵＮＩＸに於けるリアルタイム性の導入について考えていく．

6．1　構成

ＲＸ－ＵＸ８３２の構成は，図6．１に示すように1）リアルタイムカーネル，２）ファイルサーバ，３）ＵＮＩＸスーパバイザの３つのモジュールから構成されており，階層構造のアプローチを採用している．ここで，ファイルサーバとは，ＵＮＩＸカーネルのファイル機能部をＵＮＩＸのタスク管理部やメモリ管理部から切り放してモジュール化したものである．また，これよりファイルサーバとＵＮＩＸスーパバイザを併せてＵＮＩＸカーネルと呼ぶことにする．本ＯＳでは，リアルタイムカーネルを下位ＯＳとし，その上位ＯＳとしてＵＮＩＸカーネルを実現している．

また，これら各モジュールを目的に合わせて組み合わせて使用できるように設計している（図6．２）．ＵＮＩＸカーネルはＵＮＩＸＳｙｓｔｅｍＶをベースにつくられているが，リアルタイムカーネルとＵＮＩＸの結合が疎なので，ＵＮＩＸのバージョンアップに対する追従性も保持している．なお，下位ＯＳのリアルタイムカーネルには，先に当社で開発したリアルタイムＯＳ，ＲＸ６１６を採用した．

6．2　リアルタイム性の実現

6．2．1　タスクのスケジュール

リアルタイムカーネル及びＵＮＩＸカーネル内を走行するようにつくられたタスクは，リアルタイムカーネル用資源のほかにＵＮＩＸカーネル用資源を持たねばならない．リアルタイム処理のみの場合におけるオーバヘッドを最小限にするため，リアルタイム専用のタスクをつくると効率がよい．そこで，ＲＸ－ＵＸ８３２では，タスクを次のように分類した．

1）　リアルタイムタスク
　　リアルタイムカーネルのみ走行可
2）　ファイルサーバタスク
　　ファイルをアクセスできるリアルタイムタスクで，リアルタイムカーネル及びファイルサーバのみ走行可
3）　ＵＮＩＸタスク
　　全カーネル走行可

これによって，モジュール構成に対応して，タスクが存在しうることになる．

ただし，本ＯＳの構成や，これらのタスク間での通信用システムコールなどのサポートのため，タスクは一元管理するようにした．これにより，基本的にはリアルタイムカーネルのすべてのシステムコールをこれら全てのタスクが

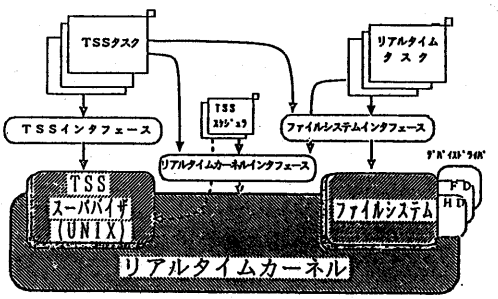
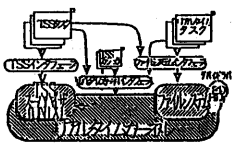
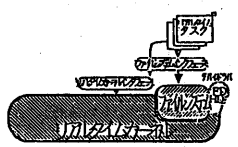


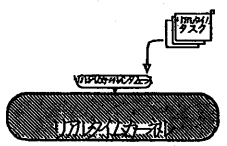
図6．1　階層構造



リアルタイム
＋
ファイル
＋
TSS



リアルタイム
＋
ファイル



リアルタイム

図6．2　コンフィグレーション

利用できるようになる．

リアルタイムカーネルにおけるタスクスケジューリングは，ユーザが予測できなければならないことから，固定優先度に従がったスケジューリングを行っており，ユーザがタスク属性としてプライオリティを指定でき，常に最高プライオリティのタスクがＣＰＵに割り付けられるようになっている．また，同一プライオリティのタスクのスケジューリングについてはラウンドロビン，メーズを選択でき，そのタイムアウト値も指定できるようにつくられている．さらに，タスクスイッチの契機としてプリエンプションをサポートしている．

ＵＮＩＸタスクのスケジューリングは，ＵＮＩＸＳｙｓｔｅｍＶのスケジューリング方式を大きく変更することなく実現している．ＵＮＩＸカーネルを走行するタスク群を図6．3のように，特定の固定プライオリティで管理し，さらにＵＮＩＸプライオリティを別に持たせて，その中でＴＳＳスケジューリングを行う．このようにして，ＵＮＩＸのスケジューリング機構をリアルタイムＯＳの機構の中に組み合わせ，これによってＵＮＩＸタスクがリアルタイムタスクの走行の妨げになったり，予測性を失わせたりしないようになっている．

しかし，ＵＮＩＸでは，タスク切り替えを禁止してＵＮＩＸカーネル内を連続走行する時間が一般的に長いため，応答性能に支障がある．そこで本ＯＳではＵＮＩＸカーネル内でのプリエンプションを許すようにした．ところが，ファイルサーバタスクもＵＮＩＸカーネル内を走行できるため，カーネル内走行の排他制御が問題になる．これについては，ＵＮＩＸカーネル内走行中は上記の固定プライオリティで走行し，ファイルサーバタスクもＵＮＩＸスケジューリングの管理下にはいり，そのあいだはタイムアウトによるタスクスイッチを行わないことで実現している（図6．3）．

6．2．2　メモリの管理

リアルタイム処理用のモジュールは，高速性，予測性の観点からメモリに常駐しており，ＵＮＩＸのメモリ管理とは性質が異なっている．また，リアルタイムカーネルは，モジュールサイズが小さくなければならないという制約があること，またＵＮＩＸのメモリ管理部分の改造を極力抑えたいことから，リアルタイムカーネルとＵＮＩＸカーネルのメモリ管理を物理，仮想とも別の空間（共有空間はリアルタイムカーネル側）に割り当てた．これによって，ＵＮＩＸカーネルの処理が過負荷状態になっても，リアルタイムカーネルのメモリ管理に影響を与えなくてすむことになる．

6．2．3　待ち状態の時間

リアルタイムカーネルでは，タスクを待ち状態に遷移させる可能性のあるシステムコールについてはそのタイムアウト値を指定できるようにすることで，走行時間の予測性を保証している．また，リアルタイムカーネルのシステム

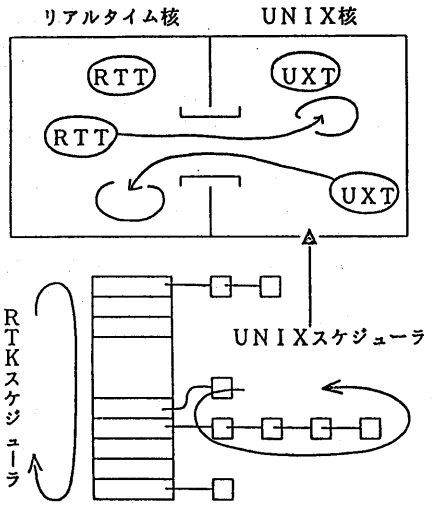


図6．3　スケジューリング

□ リアルタイム性能

• 連続ブロックファイル

□ フォールトトレラント（耐故障）機能

• 構造フリーズ
　ファイル、ファイルシステム
• 多重化ファイルシステム

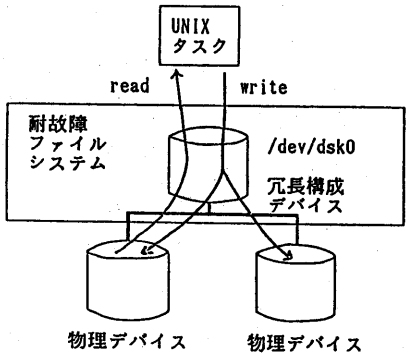


図6．4　ファイルシステムの拡張

コールは全ての種類のタスクから発行可能であるから，これらでサポートされている機能に関するリアルタイム性は保証されている．

一方，リアルタイムカーネルがサポートしていないファイルアクセス系のシステムコールは，ＵＮＩＸカーネルのものを使うことになる．従って，ＵＮＩＸカーネルのサービスを受けている間はリアルタイム性を放棄することになる．

６．２．４　ファイルの耐故障性

ファイルの耐故障性のために，ファイルシステムの多重化をサポートしている（図６．４）．これは，ファイルシステムを物理的に多重に持ち，常にバックアップファイルを維持する機能である．また，ファイルシステムの破壊を少なくするため，ファイルの内容のみの変更を許し，ファイルの生成，削除，サイズの様なファイルの管理構造の変更を許さないフリーズドファイルシステムをＵＮＩＸファイルシステムの上位互換を保ちながらサポートしている．

６．２．５　製品としての強化

ファイルアクセスに要する時間を短くするため，連続ブロックファイルをＵＮＩＸファイルシステムの上位互換を保ちながらサポートしている．これは，データブロックをデバイス内の連続域に割り当てたファイルで，ファイルの間接ブロックを使わない高速アクセスや，連続ブロックの一括アクセスが可能になる．

また，リアルタイム処理中に必ずしもＴＳＳ処理は必要ない．そこで，リアルタイム処理に対するオーバヘッドの削減のため，ＴＳＳの始動，及び停止を，リアルタイムカーネルを止めずに行えるようにしている．これは，もともとＲＸ６１６で耐故障性のためにサポートしていた再開処理機能を利用して，リアルタイム処理のみ再開させることによって実現している［４］．

また，これによりＵＮＩＸカーネルのモジュールの入れ替えも可能となる．

７．まとめ

本編では，ＵＮＩＸのリアルタイム化の導入に関する問題点と，ＲＸ－ＵＸ８３２に於ける解決法を述べてきた．

リアルタイムカーネルに上位ＯＳとしてＵＮＩＸを実現した本方式は，リアルタイムカーネルが提供するリアルタイム機能を損なうことなくＴＳＳの環境を盛り込むことを可能としている．この環境下では，リアルタイム処理とプログラム開発が並行して可能である．

最後にここでは，ＲＸ－ＵＸ８３２がこれらの方式をとったことにより，製品として強化された点を列挙する．

１）　性能

（１）スケジューリング機構
　　ＵＮＩＸカーネル内でプリエンプションを許

したため，プリエンプション応答時間はリアルタイムカーネル単体の時とほとんど変わらない．

（2）ファイルアクセス

連続ブロックファイルをサポートしたことにより，これを使用するとランダムアクセスにおいて約2倍の速度がでる（連続ブロックアクセスについては，未測定）

（3）ＴＳＳ処理の起動，停止

システムが稼働しているときに，ＴＳＳ処理部分の起動，停止を可能にしているため，リアルタイム処理に対するＴＳＳ処理のオーバヘッドの削減が可能である．

2） 信頼性，耐故障性

（1）フォルトトレラントファイルシステム

ファイルシステムの多重系に対するサポートにより，ファイルの破壊の危険性を低くした．

（2）フリーズドファイルシステム

フリーズドファイルシステムのサポートにより，ファイルシステムの破壊という致命的な破壊の危険性を低くした．

3） 柔軟なＯＳ構成が可能であること

（1）階層構造

階層構造を採用しているため，図6．1の様な形態をサポートしており，いろいろなシステムに柔軟に対応できる．

4） リアルタイムとＵＮＩＸでのシステムコールの相互利用

（1）タスク間通信

リアルタイムカーネル側でサポートしているシステムコールを，全ての種類のタスクから使用できるため，リアルタイムタスクとＵＮＩＸタスクの間で相互に同期をとったり通信をしたりできる．

（2）ファイルの共有

リアルタイム処理とＴＳＳ処理で，ファイルを共有していることにより，リアルタイム処理で収集したデータをＵＮＩＸで解析したりすることができる．

5） 開発支援環境

（1）標準ＵＮＩＸインタフェース

標準ＵＮＩＸインタフェースの採用により，標準的な環境で，豊富な流通ソフトウェアを利用することができるようになった．

これにより，アプリケーションの開発やデータ処理などをＵＮＩＸ環境で提供することによって，リアルタイムアプリケーションの開発から，テスト，デバッグまでを自身の環境で効率的に行うことができる．

参考文献

［1］下島　他：Ｖ６０／７０リアルタイムＵＮＩＸ－概要－，情処第３５回全国大会　１９８７　３Ｄ－３

［2］世良　他：Ｖ６０／７０リアルタイムＵＮＩＸ－設計－，情処第３５回全国大会　１９８７　３Ｄ－４

［3］水橋　他：Ｖ６０／７０リアルタイムＵＮＩＸ－スケジューリング方式－，情処第３５回全国大会　１９８７　３Ｄ－３

［4］古城　他：耐故障機能とランデブ機能を備えるＶ６０リアルタイムＯＳ，日経エレクトロニクス　１９８７．３．２３（ｎｏ．４１７）