4U-7　ホスト計算機上でのマルチタスク処理を支援するマルチウィンドウ端末の実現

清水謙多郎，石田晴久　(東京大学)

1．はじめに

本稿では、ワークステーション上でホスト計算機のマルチウィンドウ端末を実現し、ホスト計算機上でのマルチタスク処理を支援する通信ソフトウェア・システムＷＩＰについて述べる。ＷＩＰ（Window Interface Program）は、ホスト計算機とワークステーションの間の通信回線を多重化して用いることにより、複数の端末セッションおよびファイル転送を並行して行うことを可能にしている。ＷＩＰはユーザ・レベルのプログラムとして実現され、ワークステーション側のソフトウェアは、現在、表１に示すＯＳの異なる３機種のワークステーション上で稼動中である。

2．システムの基本構成

図１は、ＷＩＰによるホスト計算機とワークステーションとの結合を示したものである。ホスト計算機上で起動されるタスクｉに対応して、仮想端末ｉが生成され、その内容がワークステーションのディスプレイ上のウィンドウｉに表示される。ホスト計算機とワークステーションの間は物理的に１本の通信回線で結ばれているだけでよく、回線を多重化するためパケット形式でメッセージのやりとりを行う。各パケットは、メッセージがワークステーションのどのウィンドウに対応するものか、すなわちホスト計算機のどのタスクに対応するものかを識別するためのフィールドをもつ。

3．実現方式

図２は、リモート側ＷＩＰのメッセージ切換え部分をＣプログラム風に記述したものである。ここでは、通信回線および各タスクからの入力データの切換えを非封鎖入力機構を使ったポーリングにより実現している。この方法によると、メッセージの切換えは単一のタスクで実行することができる。非封鎖入力機構を用いずに実現するには、一般に、①入力データの到着を割込みで通知する、②各入力先に対応させて入力専用のタスクを起動させる（この場合、ｎ個の仮想端末を得るのに合計２ｎ＋２のタスクがホスト計算機上で稼動することになる）、といった方式が考えられる。我々は、リモート・システムとして、Ultrix（4.2bsd Unix上位互換）を搭載したVax 8600を選んだ（表１参照）が、Ultrixでは、非封鎖入力機構として、システム・コールselectを用いることができた。タスク間の通信には疑似tty（pty）を用いた。pty では、相手タスクに通常の端末と全く同じようにデータを与えることができ、vi、moreのような端末依存の

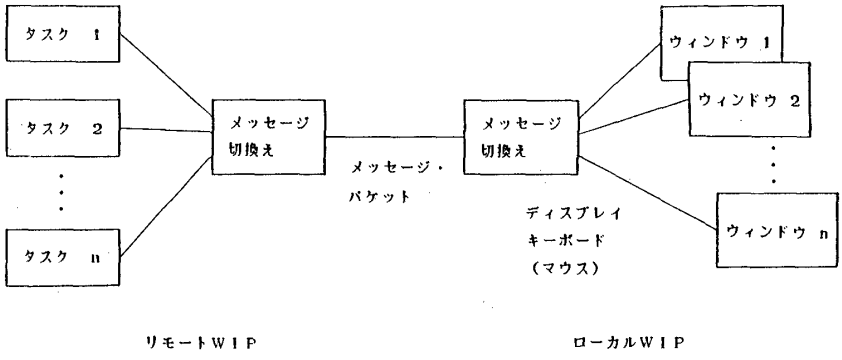

図１　ＷＩＰシステムの概略

表１　実現したシステム

マシン	ＯＳ	記述言語	プログラム・サイズ
NEC PC9801	Concurrent CP/M	C,8086アセンブリ言語	1,030 行
Hitachi 2050	HI-UX	C	950 行*
Xerox 1100SIP	Interlisp-D	Interlisp-D	570 行
Vax-8600	Ultrix-32	C	800 行

上段：　ワークステーション
下段：　ホスト・コンピュータ
*拡張機能をもつがその部分は除く

Implementation of multiwindow terminals for supporting multitasking on host computers

Kentaro SHIMIZU, Haruhisa ISHIDA

University of Tokyo

プログラムの実行をそのままサポートすることができる。なお、通常の対話セッションでは、コマンド・インタプリタのシェルがバックグラウンドで並行して実行される。

ローカル側ＷＩＰでは、仮想端末に対する入出力の切換えが行われ、その構成はリモート側とほぼ対称的である。従って、リモート側ＷＩＰと同様のメッセージ切換え方式が適用される。ただしこの場合、入力が一時に１つの仮想端末に対してのみ行われることを利用し、これを集中的に処理する方式と、仮想端末の入力をすべて調べる方式（後述するファイル転送などで適用される）が考えられる。

４．ファイル転送

ＷＩＰの拡張機能として、現在、ファイル転送機能およびホスト・システム（Ultrix）がサポートする入力行編集をローカルに行う機能が組み込まれている。これらの機能は仮想端末ごとに利用でき、メッセージ切換えタスクに専用のタスクを結合させる形で実現される。ファイル転送については、ワークステーション、ホスト計算機双方で稼動する専用のタスクの間でKermit［１］のサーバ・モードに似たプロトコルで処理が行われる。

ＷＩＰでは、先に述べた通信多重化方式により、ファイル転送中もホスト計算機と対話したり、ファイル転送自身を並行して処理することを可能にし（送信／受信両方向の並列処理が可能）、ホスト計算機との対話性を向上させている。

```
while (TRUE)
{
    if (pending input in line)
    {
        wid = receive_packet();
        write_task_input(search_task(wid));
    }
    for (i = each task)
        if (pending output in task[i]_output)
        {
            read_task_output(i);
            send_packet(task[i]_wid);
        }
}
```

図２　リモート側ＷＩＰのアルゴリズム

表２　ファイル転送多重化の効果

多重度	転送効率（bytes/sec）	比
1	87.8	1
2	157.7	1.80
2*	265.7	3.03
3	263.8	3.00
4	330.4	3.76

*双方向

また、通信路を有効に利用する上でもこうした転送の多重化は有効である。表２に、多重度によって転送効率がいかに向上するかを 9,600 bpsのRS232C回線（全二重）をもちいて測定した結果を示す。多重度３までは、ほぼ多重度に比例して転送効率が上がっている。また、両方向の転送では通信路の有効利用およびパケット処理の負荷分散により転送効率が単方向の場合に比べ向上している。

５．利用者インタフェース

ＷＩＰの実際の使用例として、図３にHitachi 2050上での画面のスナップ・ショットを示す。最後方のウィンドウはＷＩＰ立ち上げ時のウィンドウで、システム全体に関係するメッセージはここに表示される。他のウィンドウはＷＩＰにより生成されたウィンドウで、2050上のシェルが稼動中のもの（①）、ホスト計算機と交信中のもの（②，③）、ヘルプ専用ウィンドウ（④）がある。プロンプトFile> が表示されているウィンドウ（③）はファイル転送のためのウィンドウで、ワークステーション側でファイル転送を専門に行うタスクが稼動している。画面には、転送されたバイト数、パケット送出の再試行回数等が表示される。実際のウィンドウは機能別に色分けされている。

６．おわりに

ＷＩＰでは、ＯＳ、言語の異なるワークステーションに対し、通信プロトコル・レベルで移植性をもたせており、またUnixのようにＣ言語とともに広く用いられているシステムでは他機種への移植も容易と考えられる。現在、表１に示したもの以外に、Hitachi E7300、OKI if1000 にも移植済みである。我々は、また、ＷＩＰの機能をLispシステムに組み込んだ新しいLispのプログラミング環境の構築に関し研究を進めている。

参考文献

1. Cruz, F. D. and Catchings, B., "Kermit: A File-Transfer Protocol for Universities," BYTE Vol. 9, No. 6, pp.255-278 (1984).

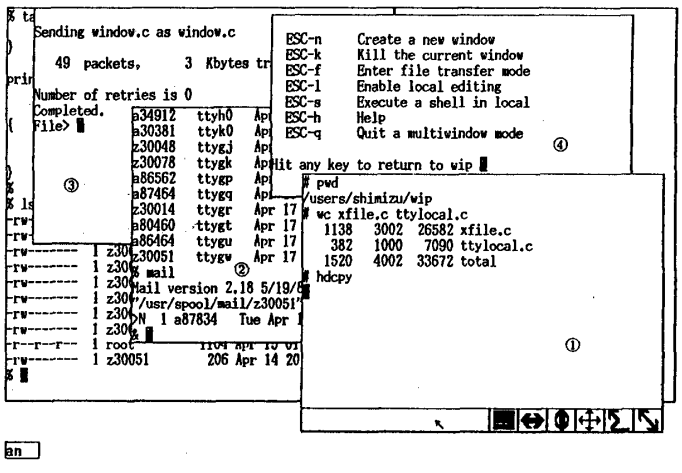

図３　Hitachi 2050上でのＷＩＰの使用例