VENIX

VENIX

VenturCom Inc.

215 First Street, Cambridge, MA 02142, USA   Tel.(617)661-1230

Table of Contents

Chapter 1

Introduction to VENIX Enhancements

This paper summarizes the major enhancements introduced in VENIX which are not found in other commercial implementations of UNIX. The paper is divided into the following sections:

- System Level Improvements

- Documentation

Chapter 2

Systems Level Improvements

2.1 Core System

Important VENIX improvements are the recoded UNIX kernel, and modifications to other key utilities. For example, the VENIX kernel resides in a single 64 kbyte memory partition in the LSI 11/23. VenturCom does not resort to "scattering" or "overlaying" the VENIX kernel as do all other UNIX vendors who offer products on micros such as the 11/23. As a result, VENIX performance on 16 bit microcomputers is superior to any other commercial UNIX implementation. The precise figures are listed below.

2.1.1 Code Size

Memory	Required By
42 to 48 kb	**VENIX Kernel** (entirely memory resident)

32 kb	basic system with DL driver
6 to 12 kb	disk and terminal buffers
1 kb	RM driver
.8 kb	per RL, RX, TM driver
.6 kb	DZ driver

2.2 Other Code Enhancements to Basic UNIX

Other items which have been improved or developed are: a **faster terminal driver** and **intelligent disk drivers**; **code mapping**; **efficient swapping algorithms**; and, improved **pipe mechanisms**.

2.2.1 Drivers

The VENIX terminal driver has been improved over standard UNIX, to allow faster I/O and provide a new control feature for examining the keyboard input buffer. This feature enables screen editors to respond immediately to a new user command without waiting to finish a previous one.

VENIX contains an improved sorting algorithm for disk drivers, providing high throughput in data transfers to disks. Since disk transfer rates are a limiting factor for most system activity, this improvement has a dramatic effect on operations.

2.2.2 Code Mapping

"Code Mapping" is a VENIX enhancement to standard UNIX which permits execution of larger than 64 kbyte programs on 16 bit computers.

Code mapping works by placing portions of the program "text" outside of the virtual address space, and by arranging each portion to be automatically mapped in and out of memory as necessary. Using this technique, the text size of VENIX programs can be nearly as large as physical memory. Data space is limited to 64kb (unless shared data segments are used as extended memory buffers as described in Section 2.3.2).

The remapping of text segments during execution is very quick and entirely automatic. Neither the user nor the programmer need be concerned with the internal mechanism. While code-mapping was designed for the DEC 11/23, it can be used advantageously on any 16 bit computer.

Other operating systems (but not standard UNIX) often provide similar functionality through "overlaying," which swaps pieces of program text to and from disks. While this approach does allow for potentially larger programs (limitations are the size of disk, not memory), overlaying is quite slow compared to code-mapping. It is our experience that the text size limitations of code-mapping are never reached. With current trends toward cheaper and cheaper memory, this situation is likely to remain true in the future.

2.2.3 Swapping Algorithm

The VENIX swapping routine is intelligent about which processes should be swapped. It makes a guess based on CPU time, priority, and I/O usage as to the nature of the process. It is heavily biased in favor of interactive programs such as the editors. Lower priority processes suffer the most.

Once a process has been swapped out, it may suffer a delay of up to several seconds before being swapped back in. If this delay is unacceptable, the process may be "locked" in memory to prevent it from being swapped out.

2.3 Real Time Processing

VENIX pioneered real-time data acquisition capabilities for UNIX and remains the only genuine commercial UNIX implementation capable of high-speed data throughput without substantial hardware support. VENIX real time features include: **Asynchronous I/O; Inter-Process Communications; Shared Data Segments; Process Timing;** and, **Pre-emptive Priorities.**

2.3.1 Asynchronous I/O

Input and output (raw or ordinary) is normally synchronous for the user; that is, after a call to "read" or "write" returns, the user is guaranteed that the transfer has been completed. If UNIX is unable to complete the transfer immediately, then it waits until resources are available, e.g. the system transfers the desired disk block into memory.

Synchronous I/O is very convenient from the programmer's point of view, and generally only a small delay is produced.

Occasionally, however, the I/O may take a significant time to complete, as with an A/D, D/A, or some remote device such as an array processor. At other times, overlapping transfers may be needed to maintain a high throughput, as when taking data from an A/D to the user's buffer and then to a disk. In these cases, "asynchronous I/O" is required.

The "asynchronous I/O" in VENIX allows a user to overlap CPU processing and any I/O which uses DMA to transfer data between the device and the user's buffer. Disk, tape and many A/D and D/A devices are in this class. The result is a dramatic increase in data throughput.

2.3.2 Inter-Process Communication & Shared Data Segments

VENIX offers the traditional UNIX inter-process communication features of "pipes", for sending information between two processes, and "signals" for asynchronously signaling a process or an event.

For applications requiring extremely fast communications between two or more processes, VENIX "shared data segments" can be used. These allow multiple processes to access a common area of memory, which can serve as a nearly instantaneous communications area. Accessing is done by windowing segments of user memory to the common area, which may be almost as large as physical memory on the computer.

A typical application for shared data segments is a process control application in which one master program must perform time-critical tasks based on requests from a number of users, and return status or diagnostic information. The master program receives its commands in the form of packets placed in the shared data segments. A unique segment area may be dedicated to each user. After performing the requested operation, the master program uses the segment to return information to the user. Both the master and user programs can read and act upon information immediately after it is placed in the segment.

Shared memory segments also can be used by individual processes as an extended memory buffer. In this mode, the segment is not "shared" at all, but serves as a convenient buffer for extremely large amounts of in-memory data. This is useful, for example, in CAD/CAM applications which must store a large amount of graphic data and access it quickly.

2.3.3 Process Timing and Pre-emptive Priority

VENIX normally allocates equal-sized slices of CPU time to each process. Users may select lower or (if permitted) higher priorities, which will bias this allocation and grant processes larger or smaller amounts of CPU time. There is also a special "pre-emptive" priority which makes all system resources exclusively available to a given process.

It sometimes is desirable for a process to suspend and resume operations for short periods of time. Standard UNIX permits processes to schedule this inactivity - to "sleep" - in increments of single seconds. VENIX has refined this capability by allowing sleeping in "clock-tick" or 1/60th second units. This is particularly useful for processes running under VENIX's special "pre-emptive" priority, as it allows a process to accurately suspend its real-time processing for short periods of time to permit normal multi-user system operation.

2.4 Communications

VENIX can communicate with other machines in two ways: as a virtual terminal emulator and via networking protocols.

2.4.1 VX - a communications utility package

The VX utility package serves both as a "virtual terminal" emulator and a transfer utility. As a virtual terminal emulator, VX allows a personal computer to login to any remote computer over a serial line, and to use the remote machine interactively. The personal computer becomes a "dumb" VT-52 type terminal, permitting editing or execution of programs on the remote machine in a natural manner.

If a user wishes to transfer files between the remote machine and a personal computer, execution of a simple command on the remote machine automatically shifts VX into "transfer mode". Data is sent back and forth in packets with full error recovery, allowing faultless transmission of files even over noisy communication lines.

"Virtual terminal" use is supported between VENIX and any remote computer with serial lines access. The file transfer capability requires support of a single utility on the remote machine. This utility is written in the C language, making it possible to use remote computers running MS-DOS, RT-11, RSX-11, VMS, HP/UX, UNIX, or other operating systems with C language compilers.

2.4.2 Networking

Microcomputer networking schemes often experience bottlenecks. With the real time features described above, VENIX has the system primitives required to achieve high throughput.

A typical networking scheme will involve three major software components:

- a device driver in the kernel;

- a network dispatcher/server in the kernel or as a user process;

- several user processes interfacing to the server.

A key objective in networking is to minimize memory-to-memory copying of information. VENIX avoids this entirely. One scenario would be as follows:

- The server tells the driver to arrange asynchronous DMA of data straight from hardware to a shared data segment controlled by the server. Note: the server is at a pre-emptive priority and locked in memory.

- The server examines the packet header and decides what to do with the information. In many cases it will be passed on to another user process.

- The server notifies (via semaphores or signals) the appropriate user process and that user process gets the information from the same shared data segment. In fact, the user process might transfer the data from shared data segment directly to disk.

Other implementations of UNIX cannot match VENIX performance in this area.

Chapter 3

Documentation

By late 1984 VenturCom will have the finest UNIX documentation on the market. We have acquired the rights to customize and reproduce a new UNIX book from McGraw-Hill which will serve as an Introduction to VENIX. Also, the tutorials in the Introduction will be delivered on-line so that a beginning user can manipulate text and data files immediately by following the key stroke examples.

We are undertaking a major rewrite of the standard UNIX manuals so that they are more readable and easily referenced. Also, we will distribute a McGraw-Hill C programming manual as part of our documentation set.

For 1985, disk based computer aided instruction is planned.