Copyright © 1983 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 62, No. 1, January 1983
Printed in U.S.A.

The 3B20D Processor & DMERT Operating System:

3B20 Field Utilities

By G. P. ELDREDGE and J. G. CHEVALIER

(Manuscript received March 10, 1982)

The term "field utilities" describes a number of tools used by telephone company craft and support staff as well as Western Electric and Bell Laboratories field support personnel for trouble-clearing and routine maintenance activities on the 3B20D/DMERT system. This complementary set of tools provides debugging coverage for the system regardless of load or system functionality. In addition, it deals with the challenges and complexities posed by the concepts of parallel processing, virtual addressing, and swapping. This article describes the various field utilities and discusses their capabilities.

I. INTRODUCTION

The term "field utilities" includes a number of tools used by telephone company, Western Electric, and Bell Laboratories support personnel to perform trouble-clearing and routine maintenance activities. Currently, software debugging and investigation tools include the Field Test Set (FTS), the Generic Access Package (GRASP), and IBROWSE, an interactive tool used to "browse" through the contents of main memory. In unusual cases, a Micro-Level Test Set (MLTS) may be used in a troubleshooting mode. The Program Documentation Standard (PDS) Field Maintenance Commands are a collection of tools used to perform more routine operational maintenance on the operating system. Each of these capabilities will be described in this article.

II. TROUBLESHOOTING AIDS

The nature of large, evolving software projects is such that, despite multiple levels of testing by developers, integration teams, system test groups, and field site acceptance teams, some software "bugs" escape

Table I—Comparison of 3B20D/DMERT debugging tools

Attribute/Tool	FTS	GRASP	IBROWSE	MLTS
Interference	None	Small, self-regulated	Small, not regulated	Extreme
Scope of capabilities	Medium	High	Low	Medium
Debugging level	Assembly	Assembly	Assembly, source	Microcode, assembly
Limitations	Limited on kernel	No special processes or kernel	No breakpoints, no trace	Difficult with supervisor or user processes, no data breakpoints
Language	C-like	PDS, MML	ADB-Like	Terse
Target users	Bell Labs, WE	Operating Co., Bell Labs, WE	Bell Labs, WE	Bell Labs, WE
Target software needed	None	DMERT	DMERT	Microcode
Support processor software needed	*UNIX* Operating System (FTS)	None	None	None
Hardware needed	FTS, DUC, terminal	UC or DUC (optional)	Terminal	MLTS, terminal
Theater of use	Limping or loaded field site	Running, nonoverloaded field site	Running, nonoverloaded field site, off-line	Lab, dead field site

detection and are included in field releases of software. In the real-time systems used in switching, the bug may be so subtle that it may surface only under equipment configurations, telephone user actions, and/or traffic loads not easily reproduced in a system laboratory environment. System debugging tools must be available in a field site carrying live traffic to solve these problems when they arise.

The 3B20D/Duplex Multiple Environment Real Time (DMERT) operating system employs advanced computer technologies that require equally sophisticated tools to isolate errors. Parallel, time-sliced execution of processes, virtual addressing, and swapping all contribute to the need for a variety and diversity of system debugging tools. Table I is a comparative summary of these various troubleshooting tools available to field sites.

2.1 Field test set

In rare cases, a system problem could occur that leaves the system

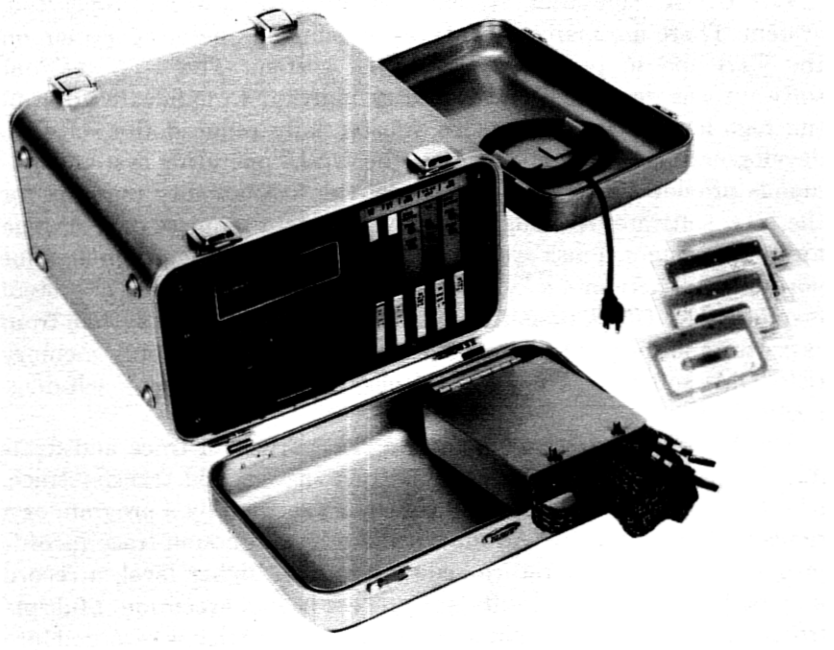

Fig. 1—Field test set.

functionally inoperative. In other cases, the traffic level may be so high that system overload mechanisms become active when unexpected results in the system indicate a software error. In either case, on-line utility systems, which assume basic functionality and nonoverload conditions, are not appropriate to isolate the problem. The Field Test Set (FTS) was designed specifically to meet this need in the field. It is strictly a monitoring device and therefore does not affect processor performance or rely on system operability. This non-interfering characteristic is extremely important when maintenance personnel are trying to isolate problems at a field site carrying a heavy traffic load.

The FTS is a small, portable unit (see Fig. 1) that is easily transported and connected to the 3B20D Processor through the Dual-Access Utility Circuit (DUC). The DUC contains hardware matchers and a 2048 entry trace memory and provides access to the processor for the FTS and GRASP (see Section 2.2). The external FTS unit connects to the DUC through an eight-foot cable. The FTS intelligence is contained in this external unit that includes a microprocessor with memory management, one megabyte of random-access memory (RAM), and a cassette transport. User access is provided through a local or remote terminal with phone access provided by the FTS.

The *UNIX** operating system was chosen as the FTS operating system. There are many advantages to using an operating system on the FTS and in particular the *UNIX* system. The FTS resident software was developed and tested as individual modules written in the high-level C language. This substantially reduced the software development time and effort. Also, the *UNIX* operating system commands provide substantial portions of the functionality required for the FTS software. Although the *UNIX* system requires disk storage for its file system, a disk system was not considered rugged enough for portability. Therefore, a "virtual disk" is supported as part of system memory. The *UNIX* operating system is booted into the system from cassette tape by resident erasable programmable read-only memory (EPROM) software. The EPROM also contains the unit's self-diagnostic software.

The FTS/DUC system supports a rich variety of trace and data-matching options. The lowest level trace, a so-called transfer trace, records program addresses of all transfers executed by a program or a range within a program. An intermediate-level function trace records program function call/return sequences. At a higher level, a record may be kept each time a different process begins execution. Multiple trace modes can be active simultaneously. Information is recorded into the trace memory under control of a variety of sophisticated matcher circuits. Masking capability is provided so that a matcher can look for a particular value of a single bit or groups of bits as well as word values. Matchers are included for address, address range, data, access type (e.g., read, write, or read/write) and process ID matching. When a matcher or a combination of matchers is triggered, a signal is produced that causes a "snap" of information into the trace memory. The matchers and matcher combinations allow very selective trace memory recording. This reduces both the size of the trace memory required and the amount of post-processing necessary to interpret the trace data.

The trace memory is operated in either a pre-trace or a post-trace mode. In the former case, the trace memory records information until it receives a stop trigger. The trace data represent program flow leading up to a particular event. In the post-trace mode, the trace memory starts recording upon receiving a trigger and stops when the trace memory is full. This provides a history of program flow after a particular event.

The DMERT operating system software is predominately written in the high-level C language. C enables the programmer to work with function-level rather than machine-level operations. To support this,

* Trademark of Bell Laboratories.

the FTS includes matching and tracing capabilities for software functions and process IDs. Function tracing records the program address and the data parameters passed on the stack to a selectable function or range of functions. Process ID matching and tracing becomes necessary in a virtual memory machine since processes are dynamically relocatable in physical memory. Processes are assigned unique ID values when they are created. The active process ID value is presented to the FTS process ID matchers and trace memory. These matchers, combined with the virtual address matchers, permit matching and tracing on virtual rather than physical addresses.

Since the FTS is an external system, it is the appropriate choice when problems must be investigated in code that has tight timing constraints or in a system that is heavily loaded. Its most attractive features are its excellent trace facility and the fact that the FTS operates in a mode that does not interfere with 3B20D operation. Although it was not designed to access machine registers or write memory, the FTS is a powerful tool in the hands of support personnel to isolate difficult system problems.

2.2 Generic access package

The concept of an on-line software debugging mechanism in real-time machines is not new.[1] Software problems may occur when the system is functional and processing traffic in a non-overload environment. Such problems can be solved in the 3B20D by use of the Generic Access Package (GRASP).

GRASP is an on-site tool for software debugging. Since it supports an interface to the DUC, GRASP provides a set of trace and data-access trap functions similar to those provided by the FTS. In addition, it provides the capability to place multiple breakpoints in code, to print the contents of memory and many machine registers, and (with some restrictions) to write memory and registers regardless of whether the DUC is available or operating correctly. GRASP has a self-regulating mechanism designed to prevent itself from taking too much real time and thereby interfering with traffic processing or driving the system into overload.

Since GRASP is "just another process" running on the machine, the design presents some unique challenges. GRASP needs to be able to identify the target process, assure that it is in main memory, and be able to gain access to its address space.

A logical process is specified by a logical tag (called a "utility ID") that is compiled into the process. All incarnations of a logical process will have the same tag since they all originate from the same object file on the disk. The tag is stored in system tables when the process is brought up and is available throughout the life of the process.

Upon a request from GRASP, the operating system searches the tables, prepares a list of real process tags (called "process IDs") for processes whose utility IDs match GRASP's request, and sends the list to GRASP. Translation between the utility ID, which is known to the craftperson, and the process ID, which is known to the operating system, is thus accomplished.

GRASP relies on cooperation with the target process to be informed when the target process is in main memory. All processes that GRASP may need to monitor must have two function calls compiled into the code, which form the run-time communication mechanism with GRASP. One is placed in the process's initial entry routine; the other is placed to execute "on demand" by GRASP.

After a process has been selected, it is forced into main memory through cooperation with the process. GRASP sends an agreed-upon event to the process; its only response to that event is to call the associated library function. That function identifies the process and notifies GRASP that it is in main memory.

Access to the target address space is then accomplished by using address translation hardware called Address Translation Buffers (ATBs). The Program Status Word (PSW) for each process is constructed to be able to handle two address spaces at one time. The identity of the address translation buffers being used by a particular process are included in that process's PSW. Instructions are provided in the instruction set to indicate which of the two address spaces to use. In addition, a special breakpoint instruction has been provided. When the breakpoint is executed by the target process, GRASP's PSW is modified so that GRASP is given access to the address space in which the breakpoint fired. This presupposes that GRASP and the target process are using different address translation buffers; that assumption is enforced by the operating system.

GRASP is especially useful when multiple breakpoints are needed (GRASP can handle up to 20), when breakpoints must be planted in several processes simultaneously, where register information is needed, or when investigation must be done remotely from a central maintenance facility.

2.3 IBROWSE

Neither the Field Test Set nor GRASP provides a mechanism to examine the kernel address space. IBROWSE, an interactive tool used only by Bell Laboratories and Western Electric support personnel, can be used to peruse the address space of any DMERT process in main memory; it fills the need to be able to view the operating system tables and message buffers in the kernel address space. IBROWSE also can be used on an off-line support processor to analyze tape dumps of main memory taken at field sites.

IBROWSE can display the contents of virtual or physical memory in a user-specified format. The user can direct that the raw machine data be represented as any combination of null-terminated strings, characters, or one-byte, two-byte (short), or four-byte (long) data types in octal, decimal, or hexadecimal format. This flexibility to specify the translation of raw data is immensely helpful when viewing DMERT data structures. IBROWSE supports the concepts of current address, next address, and current format, which are useful in displaying consecutive memory locations. It can view any process in memory, from kernel through kernel processes, supervisors, and user processes. It has the ability to search forward or backward for a specified data pattern, in either virtual or physical addressing modes. IBROWSE also supports a user-defined macro facility and I/O redirection.

The main strengths of IBROWSE are its ability to view the address space of any process in main memory and its capability to analyze data from an off-line Control Unit (CU). Since use of IBROWSE requires relatively detailed knowledge of DMERT, its users are intended to be specialized Bell Laboratories or Western Electric support personnel; for that reason, no attempt has been made to make IBROWSE part of the official DMERT release. Each time the support teams need it, they load it into the target machine.

2.4 *Micro-level test set*

Should a problem result in a "dead" system or one that is continually attempting automatic recovery actions and is unable to start the operating system, the Micro-Level Test Set (MLTS) is used. The MLTS is a low-level test system aimed primarily at hardware register and microcode access. It consists of an interface circuit that plugs into the 3B20D like any other board and an external control circuit. Since the MLTS is equipped with an RS232 interface and a 212A data set, it may be configured with a terminal on-site or may be operated from a remote location.

The MLTS is the only field utility tool that provides read/write access to all internal hardware and firmware registers and is the only one that facilitates access to the processor's microcode. The MLTS provides microcode breakpoints, can read and write microstore and main store locations, and can read and write machine registers that are not accessible to other troubleshooting tools. Although its primary use is in a laboratory environment, there are infrequent cases where such capabilities are required to solve problems during field tests.

III. OPERATIONAL UTILITIES

Since DMERT supports a hierarchical file system as well as the concept of processes, some types of problems must be dealt with and resolved at the process or file-system level. For example, a process may be running when it should not be or the file system may contain some transient files that should have been cleared. The *UNIX* operating system itself provides many utilities for process control and file system maintenance; these same capabilities are needed in the Program Documentation Standard (PDS) syntax for Electronic Switching System (ESS) applications.

PDS field maintenance commands can be described in three categories:

(*i*) File system manipulation and maintenance

(*ii*) Process control

(*iii*) Magnetic tape operations that are support-processor compatible.

PDS commands are provided to allow the craft or support person to determine what files exist on the disk and what their access permissions are; the craft may alter the access permissions, add new files, or remove existing files. A basic text editor is provided to facilitate creation or modification of ASCII files. In addition, tools are provided to start a process, stop a process, and to determine what processes are known to the system.

Although these utilities do not fall into the class of "debugging" tools, they nevertheless provide a window into the system at a high level that is very useful to solve certain types of system problems.

IV. SUMMARY

Because of its architecture and technology, the 3B20D/DMERT system presents a number of challenges to those who must isolate problems in a running system in the field. Problems may be caused by hardware failures, software deficiencies, microcode errors, or operational overloads and inconsistencies. A set of tools has been developed to isolate problems that may occur in the field. Together, these utilities provide a continuum of system trouble identification capabilities for the 3B20D/DMERT system in the field.

V. ACKNOWLEDGMENTS

The authors acknowledge the contributions of Messrs. R. H. Holt, J. P. Kehn, G. A. Moore, J. D. Peterson, and D. J. Thompson, and Ms. C. A. Toman for their inputs, critiques, and reviews.

REFERENCES

1. G. F. Clement, P. S. Fuss, R. J. Griffith, R. C. Lee, and R. D. Royer, "1A Processor: Control, Administrative, and Utility Software," B.S.T.J., *56*, No. 2 (February 1977), pp. 237–54.