1072

Bell Laboratories

Cover Sheet for Technical Memorandum

Title- Measuring UNIX

Date- October 31, 1975

TM- 75-8234-9

Other Keywords- UNIX
Computer Performance Measurement
Hardware Monitor
Benchmark

Author(s)	Location and Room	Extension
D. H. Copp	MH 2D-429	3991
P. A. Hamilton	MH 2D-426	2008

Charging Case- 70107-003
49170-110

Filing Case- 40952-001

ABSTRACT

The prime objective of this work was to evaluate three off-the-shelf tools for measuring the behavior of a UNIX system in a production environment.

The tools used were the Shell accounting feature of UNIX, and two commercial computer performance monitors ("hardware monitors"). No modifications were made to UNIX.

In order to exercise these measurement tools, and to obtain some possibly useful data, we benchmarked the UNIX operating system while running on the Department 8234 UNIX Support Group PDP-11/45, configured in four different ways. We also measured the system on a normal day.

The results published here include data on UNIX command utilization, hardware component utilization, and operating system routine utilization ("execution activity profiles").

Pages Text 12 Other 35 Total 47

No. Figures 24 No. Tables 0 No. Refs. 4

DATE FILE COPY

Address Label

SEE REVERSE SIDE FOR DISTRIBUTION LIST

BELL TELEPHONE LABORATORIES, INC.

TM-75-8234-9

DISTRIBUTION
(REFER GEI 13.9-3)

COMPLETE MEMORANDUM TO

CORRESPONDENCE FILES

OFFICIAL FILE COPY
PLUS ONE COPY FOR
EACH ADDITIONAL FILING
CASE REFERENCED

DATE FILE COPY
(FORM E-1328)

10 REFERENCE COPIES

+ARCHER,RUSSELL E JR
ARTHURS,EDWARD
+BALDWIN,GEORGE L
<BLINN,JAMES C
+BRANDT,RICHARD B
+BROWN,COLIN W
+CERMAK,IVAN A
COPP,DAVID H
<CRUM,TED A
DOWD,PATRICK G
+FABISCH,MICHAEL P
FEDER,J
<FREEMAN,K GLENN
HAMILTON,PATRICIA
<HANSEN,MRS G J
+JENKINS,J MICHAEL
JENSEN,PAUL D
<KEESE,W M
<KELLY,L J
<KERNIGHAN,BRIAN W
KLEIN,MISS R L
+LUDERER,GOTTFRIED W R
+LUDWIG,J J
+MARANZANO,JOSEPH F
<MORGAN,S P
+MUSA,J D
+NAHABEDIAN,CHARLES E
PEREZ,MRS CATHERINE D
+RALEIGH,THOMAS M
+RIDGWAY,WILLIAM C III
RITACCO,J E
+RITCHIE,DENNIS M
ROBINSON,HERMAN E
SWIFT,R E
TAGUE,BERKLEY A
+THOMPSON,K
+VOGEL,GERALD C
+WEIGLE,L W
38 NAMES

COVER SHEET ONLY TO

CORRESPONDENCE FILES

4 COPIES PLUS ONE
COPY FOR EACH FILING
CASE

COVER SHEET ONLY TO

ACKERMAN,A F
AHO,A V
AHRENS,RAINER B
ALBERTS,BARBARA A
ALCALAY,DAVID
ALLEN,JAMES R
ALLISON,CHARLES E
AMOSS,JOHN J
ANDERSON,L G
ANDERSON,MS K J
ARNOLD,GEORGE W
ARNOLD,S L
ATAL,B S
BACCASH,MISS J M
BASEIL,RICHARD J
BATNI,R P
BAUGH,C R
BAYER,DOUGLAS L
BECKER,R A
BERNSTEIN,LAWRENCE
BERTH,R P
BEYER,JEAN-DAVID
BICKFORD,N E
BILOWOS,RICHARD M
BIRCHALL,R H
BIREN,MRS IRMA B
BLUM,MRS MARION
BLY,JOSEPH A
BODEN,F J
BONANNI,LORENZO E
BOURNE,STEPHEN R
BOWERS,J L
BOWLES,J B
BOWYER,L RAY
BOYCE,W M
BROWN,JAMES W
BROWN,W STANLEY
BURES,C
BURNETTE,W A
BYRNE,EDWARD R
<CALESSO,GIULIO L
CANADAY,RUDD H
CARDOZA,WAYNE M
CARRAN,J H
CARR,DAVID C
CASPERS,MRS BARBARA E
CAVINESS,JOHN D
CHAFFEE,N F
CHAI,D T
CHAMBERS,J M
CHAMBERS,MRS E C
CHANG,HERBERT Y
CHEN,STEPHEN
CHEN,T L
CHERRY,MS L L
CHIANG,T C
CHODROW,MARK M
CHRIST,C W JR
CIESLAK,THOMAS J
CLAYTON,D P

COVER SHEET ONLY TO

CLOUTIER,J E
COBEN,ROBERT M
COHEN,HARVEY
COLE,LOUIS M
COLE,M O
COOK,THOMAS J
COREY,D A
COSTANTINO,B B
COSTELLO,PETER E
COSTON,WALTER P
COULTER,J REGINALD
CRAGUN,D W
CRANE,RODERICK P
CRUME,LARRY L
CUNNINGHAM,STEPHEN J
D STEFAN,D J
DAVIS,D R
DAVIS,R D
DE GRAAF,D A
DE JAGER,D S
+DEL RIESGO,C J
DEUTSCH,DAVID N
DI GIACOMO,J G
DICKMAN,B N
DIMMICK,JAMES O
DOLOTTA,T A
DONNELLY,MISS MARY M
DOWDEN,D C
DRUMMOND,R E
DUDLEY,MRS E H
DUNN,J C
DWYER,T J
EDELSON,D
EDMUNDS,T W
EITELBACH,DAVID L
ELY,T C
ERRICHIELLO,PHILIP M
ESSERMAN,ALAN R
ESTOCK,R G
EVERETT,W W
FACTOR,R M
FELS,ALLEN M
FISCHER,H B
FLANDRENA,R J
FLEISCHER,HERBERT I
FLYNN,MS M L
FORTNEY,V J
FOUGHT,B T
FOUNTOUKIDIS,A
FOY,J C
FRANK,H G
FRANK,MISS A J
FRANK,RUDOLPH J
FRANZ,ANN M
FRASER,A G
FREEDMAN,M I
FREEMAN,R DON
FREIDENREICH,MRS B
FROST,H BONNELL
FULTON,ALAN W
GANNON,T F

COVER SHEET ONLY TO

GARCIA,R F
GATES,G W
GAY,FRANCIS A
GEPNER,JAMES R
GERGOWITZ,E B
GEYLING,F T
GIBB,KENNETH R
GIBSON,H T JR
GIMPEL,JAMES F
GITHENS,JOHN A
GLASER,W A
GLASSER,ALAN L
GLUCK,F
GOLABEK,MISS R
GOLDSTEIN,A JAY
<GORMAN,JAMES E
GRAHAM,R L
GRAVEMAN,R F
GREENBAUM,H J
GREENE,MRS DELTA A
GREENHALGH,H WAIN
GROEGER,MRS J
GROSS,ARTHUR G
GUERRIERO,JOSEPH R
HAFER,E H
HAIGHT,R C
HALE,A L
HALFIN,SHLOMO
HALL,ANDREW D JR
HALL,JOE T
HALL,MILTON S JR
HALL,W G
HALPIN,THOMAS
HARKNESS,C J
HARRISON,NEAL T
HARTMANN,ROBERT H
HARTWELL,WALTER T
HARUTA,K
HASZTO,EDWARD D
HAUSE,A D
HEATH,SIDNEY F III
HEDRICK,MISS ELLEN L
HELD,RICHARD W
HEROLD,JOHN W
HESS,MILTON S
HESTER,S D
HOALST,BLAINE C
HOLTMAN,JAMES P
HONIG,W L
HOYT,WILLIAM F
HUANG,VICTOR K L
HUDSON,E T
<HUMCKE,D J
HUNNICUTT,CHARLES F
HUPKA,MRS FLORENCE
HYMAN,B
IERLEY,W H
IMAGNA,CLYDE P
IPPOLITI,O D
IRVINE,M M
IVIE,EVAN L

COVER SHEET ONLY TO

JACKOWSKI,D J
JACKSON,JAMES H
JACOBS,H S
JARVIS,JOHN F
JESSOP,WARREN H
JOHNSON,STEPHEN C
JOHNSTON,WALTER E JR
JONES,M B
KAMINSKY,MICHAEL L
KANE,J RICHARD
KANE,JOHN M
KAPLAN,A E
KAPLAN,M M
KAUFELD,J C JR
KAUFMAN,LARRY S
KAYEL,R G
<KETCHLEDGE,B A
KEVORKIAN,DOUGLAS E
KILLMER,JOHN C JR
KLAPPROTH,F F
KLEBER,J J
KLEINER,RICHARD T
KNUDSEN,DONALD B
KOLETTIS,N J
KORNEGAY,R L
KREUTZBERG,JOHN
LA CAVA,JOSEPH L
LAMBERT,MRS C A
LANDAU,C N
LANDOLINA,W C JR
LANG,A
LEBENHAUSEN,S
LEHMAN,WILLIAM
LEKKOS,A A
LERNER,E M
LESSER,PETER V
LESTER,THOMAS E
LEVINE,P R
LEVINSON,EDWARD
LICWINKO,J S
LIEBERT,THOMAS A
LINDERMAN,J
LIND,R O
LIST,WILLIAM H S
LONG,DAVID W
LONG,P F
LORENC,ANTHONY
LOYD,D G
LUKACS,M E
LUTZ,KENNETH J
LYCKLAMA,HEINZ
LYONS,T G
MACHOL,R E JR
MACKLER,R W
MACLENNAN,CAROL G
MACRI,PHILIP P
MADDEN,MRS D M
MALIK,JOSEPH M
MARSH,MISS M
+MARTIN,R L

...
451 TOTAL

+ NAMED BY AUTHOR > CITED AS REFERENCE < REQUESTED BY READER (NAMES WITHOUT PREFIX WERE SELECTED USING THE AUTHOR'S SUBJECT OR ORGANIZATIONAL SPECIFICATION AS GIVEN BELOW)

MERCURY SPECIFICATION.....

COMPLETE MEMO TO:
8234-AMTS

COVER SHEET TO:
COCSPM = COMPUTING SYSTEM PERFORMANCE MEASUREMENT, SIMULATION, ACCOUNTING
UNHA@ = UNIX/HARDWARE
UNSU@ = UNIX/SERVICE,UTILITY PROGRAMS

HARDISON,D
MH 2C121

TM-75-8234-9
TOTAL PAGES 45

TO GET A COMPLETE COPY:

1. BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
2. FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
3. CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.

PLEASE SEND A COMPLETE COPY TO THE ADDRESS SHOWN ON THE OTHER SIDE
NO ENVELOPE WILL BE NEEDED IF YOU SIMPLY STAPLE THIS COVER SHEET TO THE COMPLETE COPY.
IF COPIES ARE NO LONGER AVAILABLE PLEASE FORWARD THIS REQUEST TO THE CORRESPONDENCE FILES.

Table of Contents

List of Figures

Bell Laboratories

subject: Measuring UNIX
Filing Case 40952-001

date: October 31, 1975

from: D. H. Copp
P. A. Hamilton

TM-75-8234-9

MEMORANDUM FOR FILE

1. Introduction

The prime objective of this work was to evaluate three off-the-shelf tools for measuring the behavior and the performance of a UNIX system in a production environment.

The tools used were the Shell accounting feature of UNIX, and two commercial computer performance monitors ("hardware monitors").* No modifications were made to UNIX.

In order to test our measurement tools, and to obtain some possibly useful data, we benchmarked the UNIX operating system while running on the Department 8234 UNIX Support Group PDP-11/45 configured in four different ways:

RK05 root device and RF11 swap device

RP03 " " " RP03 " "

RP03 " " " RK05 " "

RP03 " " " RF11 " "

We also measured the RP03 root-RF11 swap configuration on a normal day.

The measurements were neither extensive in size nor exhaustive in scope. But they were controlled, and thus provide one objective comparison of four different UNIX hardware configurations.

*These monitors, the Dynaprobe-7900 and -8000, are manufactured by the Comress Division of Comten, Inc., and are leased by Department 8234.

2. Shell Accounting

The UNIX "Shell accounting" feature (Section VIII of [3]) was used to provide a record of command usage. The data are reported in Appendix B.

Since the ingredients of the benchmark were known a priori (see Appendix A), in the case of the benchmark runs the Shell accounting data are of interest only to determine the amount of system resources consumed. For the "normal day" measurement, the Shell accounting data play the additional role of describing the workload.

3. D-7900 Hardware Monitor

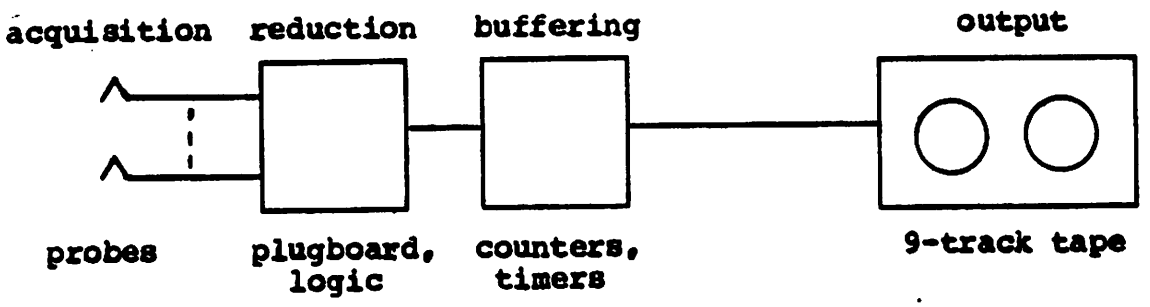

Figure 1: D-7900 Hardware Monitor

A block diagram of the D-7900 monitor, sometimes referred to as a "counter box", is given in Figure 1.

The probes are small amplifiers that attach to wire-wrap pins on the back plane of the device to be monitored. The probes are high-impedance devices, like oscilloscope probes, and do not disturb the circuits to which they are attached.

The plugboard provides access to boolean logic gates and to flip-flops, that are used to combine and process the raw signals.

The D-7900 contains 16 counter registers. Each one can be caused either to count the number of pulses in the input signal (count events), or to time-integrate the input signal (measure the fraction of time that the input signal is logical "1").

Periodically (typically, once per minute) the D-7900 writes the contents of the 16 counters, together with a time stamp, to 9-track IBM-compatible magnetic tape.

The tape is processed by a Comress-supplied data reduction program that runs on several major-vendor computer systems, including IBM 370 and Honeywell 6070.

4. D-8000 Hardware Monitor

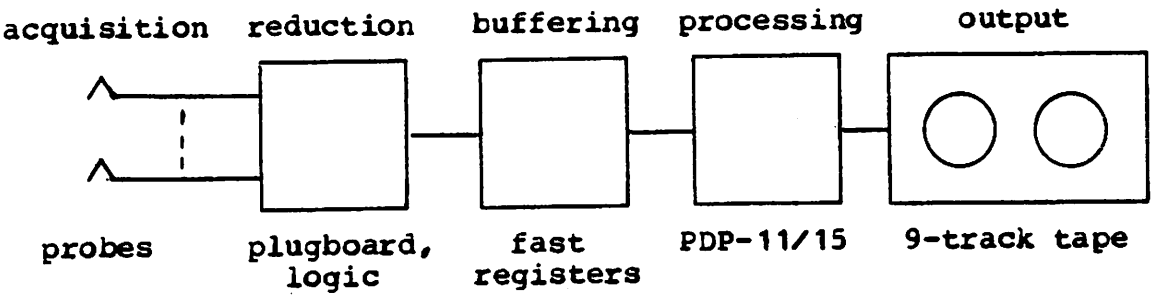

Figure 2: D-8000 Hardware Monitor

A block diagram of the D-8000 (the "profiling monitor") is given in Figure 2.

The probes are identical to the probes used by the D-7900.

The plugboard and logic are different in detail but similar in function to the plugboard and logic of the D-7900.

Whereas the D-7900 is designed to measure individual signals, the D-8000 is designed to capture whole words (groups of logically related signals, such as addresses or op-codes) of data. Thus the processed signals are buffered not in counters, but in fast (flip-flop) data buffer registers.

Processing of the data in the fast registers is performed by an integral PDP-11/15 minicomputer. The fast registers look like an I/O device to the PDP-11/15. A program in the PDP-11/15 reads the fast registers, performs some additional processing, and writes output on a 9-track tape. (The tape drive is the same one used by the D-7900, shared by means of a multiplexor not shown in the figures.)

5. Signals Monitored

D-7900 probes were attached to sufficient signals in the CPU and in the UNIBUS to monitor the following logical functions (see Appendix C for the meaning of these signals; in some cases, the names are seriously misleading):

Console lamps RUN, PAUSE, MASTER, USER and KERNEL. (SUPERVISOR was not monitored because UNIX makes no use of this processor mode.)

The UNIBUS BBSY and MSYN signals.

The READY bits in the status registers of the controllers for the RF11, RK05 and RP03 disk drives.

D-8000 probes were attached to the high-order 16 of the 18 UNIBUS (physical) address lines, and to various control lines in the UNIBUS and in the CPU that were needed to determine when to sample the UNIBUS address.

We chose to capture physical addresses rather than virtual addresses. Physical addresses appear to be easier to probe than virtual addresses. (We have not attempted to probe virtual addresses.) Physical addresses are satisfactory for profiling the UNIX operating system since it is fixed in memory.

The actual hardware probe points and the technique for finding and using them will be documented in a later memorandum.

6. Sampling

The D-7900 is fast enough that signals are in effect continuously monitored.* Because of hardware double buffering, data collection is not interrupted during writing to tape.

The D-8000, because it is controlled by a program in a minicomputer, requires about 130 microseconds to process each captured word. Thus the D-8000 can do no better than sample the UNIBUS addresses. Also, with the software we are now using, the D-8000 is inactivated during the half-second required for each tape write operation.

Many addresses appear on the UNIBUS. The CPU generates addresses when referencing instructions and data. The CPU generates a (high) address whenever an "I/O command" is executed. I/O devices generate addresses when transferring data to and from the CPU or memory.

The only addresses considered "eligible" for sampling were those generated while the CPU was bus master, i.e. only instruction fetches and CPU data fetches and stores.

The D-8000 plugboard was wired to generate a sampling pulse once every 200 microseconds (exactly). If the CPU was bus master at that instant, then the most recent eligible address was captured. Otherwise, the sampling pulse had no effect.

---

*Counting is done by counting. Time integration is done by ANDing the integrand with a 10 MHz clock and counting the pulses that survive.

7. Comments on D-8000 Sampling Method

The definition of "eligible address" included data addresses as well as instruction addresses, because we were unable to find a straightforward way to exclude the data addresses.

One consequence of this definition was to fail to attribute to a routine the samples taken while the routine was referencing data external to itself. Since the UNIX system typically loads instructions and data into distinct regions of address space ("text" and "data" segments) almost all data reference samples were excluded. (Note however that the samples taken while accessing the one or two full words that follow the instruction in certain PDP-11 addressing modes were not excluded.)

If all UNIX routines had exactly the same ratio of instruction fetch time to external data fetch time, and if all external data references were outside of all UNIX text segments, then our sampling method would underestimate all routine times by the same fraction, and the (normalized) numbers in Appendix D would be exactly correct.

Unfortunately, we do not know how much the ratio of instruction fetch time to external data fetch time varies among the routines of UNIX, but there is no reason to expect wide variation. We do know that most data reside in UNIX data segments. Thus we believe that the data in Appendix D are a good estimate of how UNIX apportions its time.

8. D-8000 Data Reduction in the PDP-11/15

We set the parameters of the program in the PDP-11/15 to cause it to construct a 3800 bar histogram (the largest possible) in its memory. Each bar in the histogram represented 16 bytes of the UNIX machine's memory, spanning from byte address 0 to byte address 166,577 octal.

Whenever a sample was captured, the PDP-11/15 added 1 to the appropriate histogram bar. Byte addresses that exceeded 166,577 octal were lumped into the topmost bar of the histogram. Thus every sample was counted in exactly one bar of the histogram.

This arrangement provided full coverage of the UNIX system (byte addresses 0 to 126,036 octal) with 16 byte resolution. (If it had been desired, we could have used the 3800 bars to span less than all of UNIX, but with higher resolution.)

Whenever any bar in the histogram (usually, the topmost bar) reached 2**16, the entire histogram was written to tape, all bars were cleared to zero, and the counting process was restarted. This happened every few minutes.

9. Off-Line Data Reduction

For each experiment, the D-7900 minute-by-minute data were printed, graphed and totalled. The totals (only) appear in Appendix C. (The minute-by-minute data and graphs are not particularly interesting.)

For each experiment, the individual D-8000 histograms were aggregated into one overall histogram spanning the entire period of the measurement. (It would have been possible to sum together only those histograms that were taken during a specific interval of the measurement.)

Consecutive 16-byte bars belonging to the same UNIX routine or data area were summed by hand. A bar that straddled two or more routines was wholly assigned to one routine, sometimes chosen by a simple rounding rule and sometimes chosen by use of judgement. (Fortunately, there were no major ambiguities in the data.) All samples not belonging to any UNIX routine, i.e. samples that fell outside of all UNIX text segments, were deleted.

The results are presented in Appendix D. Notice that the data are presented as percentages of total (undeleted) samples, i.e. as percentages of total UNIX activity, not of total CPU activity.

10. Measurements

The measurements were made at the following times:

#1 26 June, 17:25 to 18:16

#2 24 June, 17:28 to 18:21

#3 24 June, 18:26 to 19:12

#4 24 June, 16:29 to 17:15

#5 26 June, 16:28 to 17.15

#6 26 June, 09:32 to 15:29

These times are only approximate. See Figure C1 of Appendix C for the exact durations.

Figure 3 tabulates the conditions under which the measurements were made. For convenience when reading Appendices A-D, Figure 3 is repeated as Figure E1 of Appendix E.

When these measurements were made, the USG UNIX system consisted of a PDP-11/45 with 80K words of core memory. The software used for all measurements was release 3.0 of UNIX. This release separates I- and D-space. There were 15 block I/O buffers.

	#1	#2	#3	#4	#5	#6
Load	------Benchmark-------				*	Normal
Root	RK05	RP03	RP03	RP03	RP03	RP03
Swap	RF11	RP03	RK05	RF11	RF11	RF11
/usr	----------RP03----------					

*Benchmark plus "cycle sponge". See text.

Figure 3: Measurement Conditions

The benchmark used in measurements #1 - #5 is derived from a benchmark that was used to compare the PDP-11/45 with the PDP-11/70 [4]. An abbreviated listing of the version that we used is included in Appendix A.

Measurement #5 is identical to measurement #4, except that a low-priority but never-ending CPU-only background job ("cycle sponge") was added to the benchmark (see Appendix A). The purpose of this job was to soak up all excess CPU time. Comparison of the data from measurements #4 and #5 leads to some conclusions about scheduler overhead (see below).

Measurement #6 is of a "normal day's" activities. Beyond checking the system logbook, we made no attempt to verify the day's normality. (The Shell accounting data in Figure B6 of Appendix B are available for skeptics.) Measurement #6 is included to help the reader reach her or his own opinion as to how realistic the benchmark is.

For all measurements, UNIX resident routines occupied about 22K words. 58K words were available for user programs.

11. Analysis of Shell Accounting Data

See Appendix B for the data analyzed in this section.

Because the same amount of "useful work" was done for each benchmark run, we expected the USER time to be approximately constant across all benchmark runs. Figure B13 shows that the average USER time was constant to within about 1% of the mean of all benchmark runs. The fluctuation for individual commands (Figure B9) is somewhat higher; "ps" is particularly variable.

System time includes both time spent doing useful work in response to system calls from user programs, and system management overhead. The average System time did not vary by more than about 5% from the mean of all benchmark runs (Figure B13). The hardware monitor data (see next section) imply that the actual variation was on the order of 1%, and so the 5% fluctuation may be an artifact of the Shell accounting timekeeping technique. The individual commands that showed wide variation were "ps" and "pr" (Figure B10).

The amount of real time depends upon the amount of swapping, contention for I/O devices, arm positioning, etc.. It is a deficiency of the benchmark that it did not force a great deal of swapping, and so there is little variation in the averages across all benchmark runs (Figure B13). As for individual commands, only run #1 stands out: "cc" ran much slower than average, while "cat", "du" and "cp" ran much faster (Figure B8).

CPU Time and Response Time Ratio are derived from the quantities already discussed.

In Figure B14 we display the mean multiprogramming level, which is the mean number of commands concurrently in execution. For the benchmark runs this is about 4.6, reflecting the fact that there were five simulated users and essentially zero "think time". It is interesting to note that the mean multiprogramming level in run #6 is about three.

From the hardware monitor data we know that the level of activity was not constant throughout the "normal day". It would be interesting to know the mean multiprogramming level for, say, the peak hour of the day, but because Shell accounting file entries are not time-stamped we have no way to extract and analyze only those entries that pertain to a specified interval.

Because the Shell accounting records are not time-stamped, it is impossible to determine either connect time or think time from the Shell accounting data.

12. Analysis of D-7900 Data

See Appendix C for the data analyzed in this section.

The D-7900 data resemble and amplify the Shell Accounting data.

The number of seconds of User mode time (Figure C1) was within 0.5% of the mean for the first four benchmark runs. This is to be expected, because of the "equal useful work" argument. In benchmark run #5, the "cycle sponge" inflated the User mode time.

The variation in Kernel mode time was less than 1% of the mean for all five benchmark runs. We conclude that approximately equal effort is required to manage the four different resource configurations that we tested.

The CPU blocked time (see Apendix C for definition) was too small to be significant in the benchmark runs. It was even smaller (about 1% of CPU busy time--Figure C2) on the normal day. We infer that the system tended to perform CPU and I/O work alternately, rather than concurrently. This is characteristic of a low degree of multiprogramming, or of inadequate I/O buffering.

CPU wait time showed significant variation across benchmark runs. The variation in CPU wait time is the sole source of variation in benchmark duration in runs #1 - #4 (Figure C1).

Because there was almost always at least one command pending during the benchmark runs, CPU wait time must be attributed to disk arm motion and to rotational latency. We infer from the I/O data (see below) that the significant waits are for the root device, not for the swap device, a result that we did not expect. We conclude that the RK05 is a poor choice for root device (run #1).

One interesting finding is that the ratio of User to Kernel time is quite different in run #6 (the normal day) than in benchmark runs #1 - #4. On the normal day proportionately far more Kernel time was used. This is probably due to the fact that the CPU busy percentage was much lower in run #6 (27%) than in runs #1 - #4 (70% - 80%) (Figure C2)--a lightly loaded system spends proportionately more time looking for work, updating timers, etc., as may be seen in Appendix D). The minute-by-minute data for run #6 show that during periods of peak activity the ratio of User to Kernel time resembles the ratio seen in the benchmark runs.

Not RUN and PAUSE are of no intrinsic interest, but they are easy-to-measure ways to estimate CPU busy, which is not easy to measure directly. Figure C2 shows that over a wide range of CPU busy levels (27% to 98%), not RUN and PAUSE are consistently between 64% and 66% of CPU busy.

Bus MASTER and BBSY also show consistent ratios to CPU busy (Figure C3). However we would be reluctant to rely on these ratios, because we would expect deviation in the case of I/O-heavy system loads. Also, the failure of the BBSY ratio to increase from run #5 to run #6 in spite of a significant increase in the I/O busy ratio (Figure C3), and other observations, lead us to doubt that BBSY is a valid indicator of UNIBUS utilization in the PDP-11/45.

CPU MSYN is sensitive to, among other things, instruction mix. Runs #1 - #4 show similar values (Figure C3--CPU busy is the appropriate denominator), as expected. Run #5 reflects the "new" mix due to addition of the "cycle sponge". Run #6 is like runs #1 - #4.

I/O MSYN is exactly proportional to the amount of I/O traffic. The only surprise here is how low the value is for run #6 (the normal day).

The variation in I/O busy values for the benchmark runs is probably an artifact of the indirect method used to apportion UNIBUS time between CPU and I/O (see Appendix C). The time from the i-th MSYN pulse to the (i+1)-th MSYN pulse is charged to the subsystem (CPU or I/O) that issued the i-th MSYN pulse. However, if the (i+1)-th MSYN pulse does not occur within a few microseconds, the D-7900 is wired to "time-out" and the UNIBUS is declared idle. In runs #1 -#4 and #6 time-outs are common, and time-outs produce an over-estimate of UNIBUS usage. In run #5, because of the "cycle sponge", there should have been almost no time-outs.

Runs #4 and #5 should have produced the same amount of I/O busy time; the drastic difference in measured values (90 sec. vs. 58 sec.) indicates that the I/O busy measuring technique is not trustworthy.

The I/O device busy times are surprising. For all runs except #2, for which the data are not available because the same device was used for root as for swap, the root device was busy for about ten times as long as the swap device (Figure C1). A very bad root file layout could cause high root device time, but we doubt that this was the cause of what we observed.

Finally, we do not know why there was significantly less swapping in run #4 than in runs #1 and #5, as shown by the RF11 busy times and the RF11 command counts in Figure C1.

13. Analysis of D-8000 Data

See Appendix D for the data analyzed in this section.

The memory profiles (Figure D1) show how much time was spent in each UNIX routine. We expected to find the first four runs to be quite similar, with the exception of the device handlers, and it can be noted that this was the case.

Figure D2 lists the twenty-two most time-consuming routines on the normal day (run #6). _swtch heads the list; it is invoked every time a process is switched. Time is also spent in _swtch while waiting for "useful work"; this is illustrated by the run with the "cycle sponge" (run #5), which had the least time in _swtch. The highest (proportional) utilization of _swtch occurred on the normal day, which had the lowest CPU utilization (Figure C2). Among the benchmark runs, the slightly larger amount in run #1 is probably due to the slightly lower CPU utilization (Figure C2).

Several of the routines listed in Figure D2 reflect the fact that on the normal day the ratio of Kernel mode to User mode time was high (see Figure C2): _wakeup and _clock (both associated with system functions) showed significantly higher utilization on the normal day than in the benchmark runs, while csv and cret, _trap, _fuword, _suword, _sureg, _copyseg and _rdwr (all associated with user calls to the system) showed significantly lower utilization

on the normal day.

The benchmark does not use the DH multiplexer (_dhxint, _dhstart), which showed moderate usage in the normal day run. Similarly, all terminal output was suppressed in the benchmark runs (_ttyoutp).

_getblk usage varied by configuration. It appears that when the RK05 is configured either as the swap or root device, _getblk usage drops significantly. We do not know why _getblk usage was so low on the normal day.

The memory profiles did not show any major deviations from what we had expected, however this does not mean that the profiles were valueless. They confirm that the system was running normally, free from unsuspected performance bugs. This is useful information.

14. Conclusions

The average CPU utilization on the "normal day" was 27%. The USG system is considered to be fairly heavily loaded, however this measurement indicates that the CPU is probably not the bottleneck.

The difference in duration across benchmark runs was almost wholly due to the difference in CPU wait times. The nature of the benchmark assures us that all CPU wait time was spent waiting for disks.

Unibus contention was unimportant.

In all runs, including the "normal day", the root device was used ten times as much as the swap device. Thus we conclude that reducing the traffic to, or speeding up, the root device, is the key to improving the performance of the USG UNIX system.

Our conclusions concerning the prime objective of the experiment (see the Introduction) are as follows:

Shell accounting proved to be a productive and easy-to-use tool. Hardware monitor measurements verify that the Shell accounting data are accurate. The statistics gathered by Shell accounting are sufficient for overall comparisons of different system configurations. The usefulness of these data would be increased by including a time-stamp with each command.

The D-7900 provided information about the utilization of the various hardware components of the system. These data are useful for finding bottlenecks. We still lack a reliable method for measuring I/O busy time.

The D-8000 provided information about the utilization of the various software components of the system. These data are of interest mainly to those responsible for maintaining and modifying the system.

In summary, Shell accounting is a good tool for measuring the level of UNIX performance. Hardware monitoring can help to explain why that level of performance is attained.

15. Acknowledgements

We wish to thank the entire UNIX Support Group for their assistance. In particular, G. C. Vogel first suggested the project, J. F. Maranzano advised us in the design of the experiment, R. B. Brandt supplied the benchmark and helped us to run it, and T. M. Raleigh served as our UNIX internals consultant.

16. References

1. PDP-11/45 Processor Handbook, Digital Equipment Corporation, Maynard, Massachusetts, 1972.

2. PDP-11 Peripherals Handbook, 1973-74, Digital Equipment Corporation, Maynard, Massachusetts, 1973.

3. UNIX Programmer's Manual, Fifth Edition, by K. Thompson and D. M. Ritchie, Bell Telephone Laboratories, 1974.

4. Benchmark of UNIX on DEC 11/70, by R. B. Brandt and K. Thompson, TM 75-8234/1271-4, April 17, 1975.

D H Copp

D. H. Copp

P A Hamilton

P. A. Hamilton

MH-8234-DHC-dhc
-PAH-

Attachments

Appendix A: The Benchmark
Appendix B: Shell Accounting Data
Appendix C: D-7900 Data
Appendix D: D-8000 Data
Appendix E: Measurement Conditions

Figures: A1
B1-B14
C1-C3
D1-D2
E1

APPENDIX A: The Benchmark

Count and Relative Frequency

ed	90	17.18%
sh	60	11.45
ls	60	11.45
cat	50	9.54
rm	40	7.63
cp	30	5.73
cc	20	3.82
pr	20	3.82
du	20	3.82
mv	20	3.82
**gok	10	1.91
colv	10	1.91
ps	10	1.91
grep	10	1.91
pwd	10	1.91
rmdir	10	1.91
who	10	1.91
echo	10	1.91
mkdir	10	1.91
chmod	10	1.91
date	10	1.91
sleep	4	0.76
Total	524	

Notes: In the benchmark, "**gok" is an execution of nroff and "colv" is a program that resembles "gsi".

Figure A1: Benchmark Command Distribution

Running the benchmark consists of asynchronously executing five copies of a script. The copies are started 15 seconds apart by use of "sleep" commands, and the benchmark is over when all copies have completed. All I/O is to files.

Each copy of the script consists of 104 UNIX commands. (Note: Each invocation of "ed" counts as one command.) The command mix was chosen to resemble the command mix seen in normal operation of the UNIX Support Group and the Department 9152 UNIX systems (except for the addition of "date" commands that were necessary to establish the start and stop times of each copy of the script). However there was no attempt to match the characteristics of the individual commands, i.e. no attempt to make the script "ed" commands equal in duration to the average real-world

"ed" command, etc.. Also, the benchmark does not contain any "think" time--each command is issued immediately after the previous command completes.

The benchmark is a modified version of the one used to compare the PDP-11/45 with the PDP-11/70 in February 1975 [4].

The command mix of the benchmark may be seen in Figure A1. An abbreviated listing of the benchmark itself is presented below; long bodies of text, included to exercise the editor, have been deleted to save space. The full benchmark is available from the authors.

The command to execute the benchmark is:

```
sh shell5  2  15
```

where "2" is a version number and "15" is the number of seconds to sleep between initiations of the script.

During run #5 only, there was a background "cycle sponge" job in the system throughout the run. This job consisted of the one-instruction loop

```
BR .
```

The "nice" command was used to give the "cycle sponge" low priority. Because of its small size, we believe that it was only rarely (if ever) swapped out, however we have no verification of this.

Abbreviated Listing of Benchmark

shell5

```
time sh shscript$1 a  &
sleep $2
time sh shscript$1 b  &
sleep $2
time sh shscript$1 c  &
sleep $2
time sh shscript$1 d  &
sleep $2
time sh shscript$1 e  &
```

shscript2

```
echo UNIX Script Run $1
date
pwd >junk1$1
who >junk1$1
ls -lu >junk3$1
```

```
sh sh6 $1
ls -l >>junk3$1
ed - junk3$1
1,$s/$/end of line/
1,$s/^/start of a line/
1,3m$
w
q
cat junk1$1 junk2$1 junk3$1 >junk4$1
sh sh1 $1
sh sh2 $1
rm junk?$1
mkdir sc$1
chdir sc$1
cp ../dummy dte.c
ed - dte.c
$a
```

(3 pages of text (C program) deleted)

```
.
1d
w
q
ed - ../srce1.c
$a
Z
.
$r ../srce2.c
w two.c
$r ../srce3.c
w three.c
/Z/,$d
w one.c
q
cat one.c two.c three.c >all.c
cc dte.c
cp all.c junk1$1
sh ../sh3 $1
cp ../srce4.c s1.c
cc s1.c
rm *
chdir ..
rmdir sc$1
sh sh4 $1
rm junk3$1
sh sh5 $1
ls -lu >junk6$1
rm junk?$1
pwd >junk1$1
who >junk1$1
ls -lu >junk3$1
sh sh6 $1
ls -l >>junk3$1
```

```
ed - junk3$1
1,$s/$/end of line/
1,$s/^/start of a line/
1,3m$
w
q
cat junk1$1 junk2$1 junk3$1 >junk4$1
sh sh1 $1
sh sh2 $1
rm junk?$1
mkdir sc$1
chdir sc$1
cp ../dummy dte.c
ed - dte.c
$a
```

(3 pages of text (C program) deleted)

```
.
1d
w
q
ed - ../srce1.c
$a
Z
.
$r ../srce2.c
w two.c
$r ../srce3.c
w three.c
/Z/,$d
w one.c
q
cat one.c two.c three.c >all.c
cc dte.c
cp all.c junk1$1
sh ../sh3 $1
cp ../srce4.c s1.c
cc s1.c
rm *
chdir ..
rmdir sc$1
sh sh4 $1
rm junk3$1
sh sh5 $1
ls -lu >junk6$1
rm junk?$1
echo End Script $1
date
```

sh1

```
ps -alx >junk5$1
pr junk1$1 >>junk5$1
ed - junk5$1
v/m/s/^/t/
1,/t/+4s/t/T/
w
q
```

sh2

```
du -a >junk6$1
ed - junk6$1
$a
1
4
.
1,$s/1/2/
1,$s/4/3/g
3r srce5.c
w
q
```

sh3

```
chmod 0707 a.out
ls -l >junk2$1
du -a .. >junk3$1
cat junk2$1 junk3$1 >junk4$1
mv junk1$1 junk5$1
```

sh4

```
ls >junk1$1
ed - junk1$1
v/sh/d
w
q
mv junk1$1 junk3$1
cat srce5.c junk3$1 >junk1$1
pr junk1$1 >junk4$1
```

sh5

```
ed - junk1$1
1,$c
```

(15 1/2 pages of text (C program) deleted)

```
.
w
q
ls -l junk1$1 >>junk2$1
ed - junk2$1
$r srce1.c
1,$s/a/a/
1,$s/ / /g
w
q
cat junk[1-4]$1 >junk5$1
```

sh6

```
nroff pidentxmac pidentx ^ colv >junk1$1
grep COMPILE junk1$1 >junk2$1
ed - junk2$1
1,$s/COMPILE/compile/
0a
this is the result of a grep on
a nroffed file.
.
w
q
```

APPENDIX B: Shell Accounting Data

The UNIX Shell accounting feature (Section VIII of [3]) records for each command executed the user time, system time and real (wall-clock) time consumed, and the user ID. Note that the time of execution ("time-stamp") is not recorded.

There is a UNIX command, "sa", that summarizes the Shell accounting file. (Warning to users of "sa": the data in the accounting summary file are also included.) We have used this command, with options c, j, l and t, to produce the figures in this appendix. (For some reason, you omit option a if and only if you include the letter "a" in the argument list.)

The normal day's data were analyzed by the standard UNIX "sa" command, while the data from the benchmark runs were analyzed by a special version of "sa". In both versions of "sa", the time consumed by subcommands of a Shell procedure is attributed to the "sh" command that invoked the procedure. In the standard version of "sa" there is no other accounting for the subcommands,* however in the special version the subcommands are counted again under their own names. Thus in the special version, subcommand times are counted twice.

Due to a bug, the Shell accounting feature randomly loses a few percent of all commands.* Thus the "Count" integers in the figures are low. Figure B7 highlights the effects of this bug. However because (so far as we know) all command types are roughly equally likely to be lost, the "Count" percentages and the time values in the figures should not be significantly affected by this random data loss.

Also notice in Figure B7 that there is an extra "sh", "cp" or "mv" command in one or more of the runs. These commands were manually executed in connection with initializing and stopping the benchmark runs. Due to an oversight we failed to conceal them from our data collection tools.

*This may be changed in the future.

	Count		Real Time		USER Time		System Time		RTR
Summary	518		40.81		3.05		2.81		7.0
sh	61	11.78%	110.00	31.74%	9.78	37.82%	9.49	39.76%	5.7
ed	86	16.60%	45.73	18.61%	1.51	8.21%	5.43	32.08%	6.6
cc	20	3.86%	291.65	27.59%	14.70	18.64%	7.05	9.69%	13.4
**gok	10	1.93%	157.90	7.47%	27.88	17.67%	2.81	1.93%	5.1
colv	10	1.93%	1.10	0.05%	12.99	8.24%	1.39	0.95%	0.1
pr	20	3.86%	24.45	2.31%	2.98	3.77%	2.99	4.11%	4.1
ls	59	11.39%	11.81	3.30%	0.96	3.58%	0.68	2.74%	7.2
cat	50	9.65%	7.36	1.74%	0.31	0.97%	0.65	2.23%	7.7
du	20	3.86%	8.00	0.76%	0.07	0.09%	1.20	1.64%	6.3
ps	9	1.74%	58.56	2.49%	0.06	0.04%	2.41	1.49%	23.6
grep	10	1.93%	13.00	0.61%	1.43	0.91%	0.79	0.54%	5.8
rm	40	7.72%	5.85	1.11%	0.01	0.04%	0.41	1.12%	13.8
cp	29	5.60%	4.59	0.63%	0.00	0.01%	0.38	0.76%	11.8
pwd	10	1.93%	6.10	0.29%	0.00	0.00%	0.37	0.26%	16.3
mv	21	4.05%	3.00	0.30%	0.00	0.00%	0.14	0.20%	21.4
rmdir	10	1.93%	2.70	0.13%	0.00	0.00%	0.17	0.12%	16.0
mkdir	10	1.93%	3.40	0.16%	0.00	0.00%	0.13	0.09%	26.5
who	9	1.74%	2.78	0.12%	0.00	0.00%	0.11	0.07%	24.2
date	10	1.93%	1.40	0.07%	0.01	0.00%	0.09	0.06%	14.0
echo	10	1.93%	1.80	0.09%	0.00	0.00%	0.09	0.06%	19.6
chmod	10	1.93%	2.90	0.14%	0.00	0.00%	0.08	0.06%	32.8
sleep	4	0.77%	16.00	0.30%	0.00	0.00%	0.10	0.03%	167.0
Summary	510		39.59		3.10		3.03		6.5
sh	59	11.57%	120.27	35.14%	10.21	38.04%	10.49	40.07%	5.8
ed	90	17.65%	42.79	19.07%	1.46	8.32%	5.50	32.04%	6.1
cc	18	3.53%	192.89	17.19%	15.97	18.16%	7.56	8.81%	8.2
**gok	10	1.96%	158.40	7.84%	28.14	17.78%	2.75	1.78%	5.1
colv	10	1.96%	1.10	0.05%	13.03	8.23%	1.43	0.93%	0.1
pr	17	3.33%	32.06	2.70%	3.49	3.74%	2.51	2.76%	5.3
ls	57	11.18%	13.95	3.94%	1.00	3.58%	0.70	2.60%	8.2
ps	10	1.96%	48.00	2.38%	0.20	0.12%	6.56	4.25%	7.1
cat	50	9.80%	13.24	3.28%	0.31	0.98%	0.65	2.09%	13.9
du	19	3.73%	28.63	2.69%	0.06	0.07%	1.27	1.56%	21.6
grep	10	1.96%	17.40	0.86%	1.39	0.88%	0.77	0.50%	8.0
rm	40	7.84%	8.25	1.63%	0.02	0.04%	0.42	1.08%	19.1
cp	28	5.49%	9.32	1.29%	0.01	0.01%	0.36	0.66%	25.1
pwd	10	1.96%	7.80	0.39%	0.01	0.01%	0.36	0.23%	21.4
mv	18	3.53%	3.78	0.34%	0.01	0.01%	0.13	0.15%	27.2
rmdir	10	1.96%	3.30	0.16%	0.00	0.00%	0.17	0.11%	19.4
mkdir	10	1.96%	5.00	0.25%	0.00	0.00%	0.14	0.09%	34.9
who	10	1.96%	3.60	0.18%	0.01	0.01%	0.12	0.08%	27.7
date	10	1.96%	1.30	0.06%	0.01	0.01%	0.10	0.06%	12.0
echo	10	1.96%	2.00	0.10%	0.01	0.00%	0.10	0.06%	19.0
chmod	10	1.96%	2.10	0.10%	0.00	0.00%	0.08	0.05%	25.2
sleep	4	0.78%	17.25	0.34%	0.00	0.00%	0.12	0.03%	142.8

Figures B1 and B2: RK05 root-RF11 swap; RP03 root-RP03 swap

	Count		Real Time		USER Time		System Time		RTR
Summary	506		37.88		3.08		2.96		6.3
sh	59	11.66%	116.83	35.96%	10.13	38.28%	10.30	40.52%	5.7
ed	82	16.21%	43.70	18.69%	1.42	7.46%	5.78	31.59%	6.1
cc	19	3.75%	177.63	17.61%	15.29	18.61%	7.45	9.44%	7.8
**gok	10	1.98%	152.60	7.96%	27.63	17.70%	2.82	1.88%	5.0
colv	10	1.98%	0.80	0.04%	13.00	8.33%	1.40	0.94%	0.1
ls	58	11.46%	11.59	3.51%	0.96	3.56%	0.70	2.71%	7.0
pr	20	3.95%	27.95	2.92%	3.02	3.86%	1.45	1.93%	6.3
ps	9	1.78%	46.11	2.16%	0.26	0.15%	6.50	3.90%	6.8
cat	47	9.29%	12.89	3.16%	0.33	0.99%	0.65	2.04%	13.2
du	20	3.95%	25.00	2.61%	0.06	0.07%	1.32	1.76%	18.1
grep	10	1.98%	17.80	0.93%	1.42	0.91%	0.74	0.49%	8.3
rm	39	7.71%	6.77	1.38%	0.02	0.05%	0.44	1.13%	14.9
cp	30	5.93%	8.50	1.33%	0.01	0.01%	0.37	0.73%	22.7
pwd	10	1.98%	5.90	0.31%	0.01	0.00%	0.36	0.24%	15.9
mv	19	3.75%	3.32	0.33%	0.00	0.00%	0.14	0.18%	22.6
rmdir	10	1.98%	3.60	0.19%	0.00	0.00%	0.15	0.10%	23.0
mkdir	10	1.98%	4.20	0.22%	0.01	0.00%	0.15	0.10%	27.4
who	10	1.98%	2.20	0.11%	0.01	0.00%	0.12	0.08%	17.4
date	10	1.98%	1.50	0.08%	0.00	0.00%	0.10	0.07%	14.1
echo	10	1.98%	1.10	0.06%	0.00	0.00%	0.10	0.07%	10.5
chmod	10	1.98%	2.00	0.10%	0.00	0.00%	0.08	0.06%	22.6
sleep	4	0.79%	16.75	0.35%	0.00	0.00%	0.08	0.02%	223.3
Summary	514		37.55		3.07		3.03		6.1
sh	61	11.87%	112.41	35.52%	9.89	38.19%	10.31	40.33%	5.6
ed	87	16.93%	42.86	19.32%	1.40	7.73%	5.66	31.55%	6.1
cc	20	3.89%	175.35	18.17%	14.92	18.89%	7.30	9.36%	7.9
**gok	10	1.95%	152.50	7.90%	27.91	17.67%	2.80	1.79%	5.0
colv	10	1.95%	1.90	0.10%	13.06	8.27%	1.39	0.89%	0.1
pr	19	3.70%	29.32	2.89%	3.14	3.78%	2.55	3.10%	5.2
ls	56	10.89%	10.93	3.17%	0.94	3.34%	0.68	2.44%	6.7
ps	10	1.95%	40.30	2.09%	0.19	0.12%	6.02	3.86%	6.5
cat	47	9.14%	12.06	2.94%	0.32	0.95%	0.65	1.97%	12.4
du	20	3.89%	25.05	2.60%	0.06	0.08%	1.22	1.57%	19.5
grep	10	1.95%	15.60	0.81%	1.41	0.89%	0.78	0.50%	7.1
rm	39	7.59%	6.74	1.36%	0.02	0.04%	0.42	1.04%	15.6
mv	21	4.09%	3.29	0.36%	0.00	0.00%	0.14	0.19%	23.1
cp	31	6.03%	8.29	1.33%	0.01	0.02%	0.36	0.72%	22.2
mv	21	4.09%	3.29	0.36%	0.00	0.00%	0.14	0.19%	23.1
rmdir	10	1.95%	3.30	0.17%	0.00	0.00%	0.17	0.11%	19.6
who	10	1.95%	2.80	0.15%	0.01	0.01%	0.14	0.09%	19.1
mkdir	10	1.95%	4.70	0.24%	0.00	0.00%	0.13	0.08%	37.1
echo	10	1.95%	1.20	0.06%	0.00	0.00%	0.10	0.07%	11.4
date	10	1.95%	1.10	0.06%	0.01	0.01%	0.09	0.06%	11.4
chmod	9	1.75%	2.11	0.10%	0.00	0.00%	0.07	0.04%	30.8
sleep	4	0.78%	16.75	0.35%	0.00	0.00%	0.10	0.03%	160.8

Figures B3 and B4: RP03 root-RK05 swap; RP03 root-RF11 swap

	Count		Real Time		USER Time		System Time		RTR
Summary	501		38.70		3.13		3.03		6.3
sh	60	11.98%	116.27	35.98%	10.14	38.78%	10.22	40.47%	5.7
ed	84	16.77%	43.67	18.92%	1.48	7.94%	5.83	32.28%	6.0
cc	19	3.79%	176.89	17.34%	14.41	17.44%	7.41	9.29%	8.1
**gok	10	2.00%	162.40	8.38%	28.49	18.16%	2.73	1.80%	5.2
colv	10	2.00%	1.50	0.08%	13.09	8.34%	1.38	0.91%	0.1
ls	59	11.78%	11.95	3.64%	0.98	3.67%	0.67	2.62%	7.2
pr	19	3.79%	26.74	2.62%	2.91	3.52%	1.78	2.24%	5.7
ps	9	1.80%	45.89	2.13%	0.21	0.12%	6.18	3.67%	7.2
cat	48	9.58%	12.40	3.07%	0.32	0.99%	0.64	2.02%	12.9
du	19	3.79%	25.95	2.54%	0.04	0.05%	1.29	1.61%	19.5
grep	10	2.00%	18.30	0.94%	1.39	0.88%	0.79	0.52%	8.4
rm	39	7.78%	7.23	1.45%	0.02	0.06%	0.42	1.07%	16.4
cp	29	5.79%	8.55	1.28%	0.01	0.01%	0.38	0.72%	22.3
pwd	8	1.60%	6.75	0.28%	0.01	0.00%	0.37	0.20%	17.9
mv	19	3.79%	3.11	0.30%	0.01	0.01%	0.11	0.14%	27.0
who	10	2.00%	2.60	0.13%	0.01	0.00%	0.13	0.09%	18.6
rmdir	8	1.60%	4.13	0.17%	0.01	0.00%	0.17	0.09%	23.9
echo	10	2.00%	1.30	0.07%	0.01	0.01%	0.11	0.07%	10.8
mkdir	8	1.60%	4.63	0.19%	0.00	0.00%	0.13	0.07%	34.2
date	9	1.80%	1.00	0.05%	0.01	0.01%	0.09	0.05%	9.5
chmod	10	2.00%	1.90	0.10%	0.01	0.00%	0.07	0.05%	24.8
sleep	4	0.80%	16.75	0.35%	0.00	0.00%	0.10	0.03%	167.5

Figure B5: RP03 root-RF11 swap, "cycle sponge" added

In the figures, the columns have the following meanings:

Count — The number of instances of the command, and the relative frequency of the command.

Real Time — The mean number of seconds of wall-clock time per instance of the command, and the percent of the sum of all real time values due to all instances of this command (note that the sum of all real time values typically exceeds the total elapsed time).

USER Time — The mean number of seconds of CPU USER-mode time per instance of the command, and the percent of all CPU USER-mode time consumed by all instances of the command.

System Time — The mean number of seconds of CPU KERNEL-mode time per instance of the command, and the percent of all CPU KERNEL-mode time consumed by all instances of the command.

RTR — RTR, the response time ratio, is the ratio of the mean Real Time to the mean CPU (USER+System) time.

The Summary line gives the total number of commands, and weighted averages for the other quantities. "sa" computes the weighted average times by dividing the total amount of time by the total number of commands, and the weighted average RTR by dividing the total accumulated real times by the total CPU time.

When command "x" is timed by the "time" command, Shell accounting charges the real, user and system time consumed both to command "x" and to command "time". The "time" command was invoked five times during run #6, and, unfortunately, was used to time commands that consumed a large fraction of the system's resources. To avoid seriously distorting the percentages, we deleted all "time" commands from the Shell accounting data before preparing Figure B6.

The commands are sorted by decreasing total CPU (USER+System) time.

Figure B6 begins here.

	Count	Real Time		USER Time		System Time		RTR
Summary	930	77.33		1.18		1.72		26.7
ed	202	230.13	64.64%	1.36	25.03%	1.12	14.21%	92.7
**gok	20	9.00	0.25%	4.89	8.92%	17.55	21.96%	0.4
nroff	69	72.91	7.00%	3.99	25.11%	1.68	7.24%	12.9
cc	21	33.71	0.98%	11.26	21.57%	4.41	5.79%	2.2
dump	1	843.00	1.17%	13.58	1.24%	161.23	10.09%	4.8
ics	3	770.33	3.21%	9.24	2.53%	40.88	7.67%	15.4
sh	30	36.10	1.51%	0.53	1.45%	4.01	7.53%	7.9
test	4	138.00	0.77%	6.28	2.29%	19.96	4.99%	5.3
cp	31	9.77	0.42%	0.03	0.08%	1.76	3.42%	5.5
a.out	12	18.08	0.30%	3.43	3.76%	0.79	0.60%	4.3
check	1	187.00	0.26%	22.22	2.03%	20.83	1.30%	4.3
ls	138	4.52	0.87%	0.07	0.92%	0.21	1.85%	15.8
ps	6	27.83	0.23%	0.20	0.11%	5.44	2.04%	4.9
wump	4	785.00	4.37%	0.68	0.25%	7.05	1.76%	101.6
sky	6	39.00	0.33%	3.62	1.98%	0.84	0.31%	8.7
e	2	3038.00	8.45%	2.17	0.40%	8.45	1.06%	285.9
lpr	17	3.00	0.07%	0.08	0.13%	0.86	0.92%	3.2
as	22	5.23	0.16%	0.04	0.09%	0.65	0.89%	7.5
man	8	27.75	0.31%	0.14	0.10%	1.22	0.61%	20.5
find	4	7.25	0.04%	0.09	0.03%	2.55	0.64%	2.7
cat	66	19.89	1.83%	0.01	0.07%	0.14	0.58%	131.5
pr	21	17.38	0.51%	0.06	0.12%	0.37	0.48%	40.7
cal	22	5.64	0.17%	0.03	0.06%	0.35	0.48%	14.9
111	1	13.00	0.02%	5.22	0.48%	2.80	0.18%	1.6
rm	47	1.23	0.08%	0.00	0.01%	0.16	0.46%	7.7
nm	11	3.73	0.06%	0.35	0.35%	0.21	0.15%	6.6

mail	10	18.00	0.25%	0.03	0.03%	0.53	0.33%	31.9
pwd	15	1.13	0.02%	0.01	0.01%	0.37	0.34%	3.0
linos	2	4.50	0.01%	1.98	0.36%	0.80	0.10%	1.6
db	11	56.36	0.86%	0.01	0.01%	0.37	0.25%	151.2
od	4	21.25	0.12%	0.38	0.14%	0.59	0.15%	22.0
cref	4	2.75	0.02%	0.11	0.04%	0.61	0.15%	3.8
gsi	1	180.00	0.25%	0.62	0.06%	2.08	0.13%	66.7
nfs	4	7.25	0.04%	0.14	0.05%	0.52	0.13%	11.0
mv	13	1.23	0.02%	0.00	0.00%	0.18	0.15%	6.5
who	10	7.50	0.10%	0.04	0.03%	0.16	0.10%	37.5
usum	1	39.00	0.05%	0.12	0.01%	1.77	0.11%	20.7
grep	4	1.00	0.01%	0.14	0.05%	0.25	0.06%	2.6
ld	7	1.00	0.01%	0.00	0.00%	0.19	0.08%	5.1
dpr	2	3.50	0.01%	0.08	0.02%	0.50	0.06%	6.0
stat	1	6.00	0.01%	0.37	0.03%	0.72	0.04%	5.5
bj	1	68.00	0.09%	0.03	0.00%	0.85	0.05%	77.0
execute	7	0.86	0.01%	0.02	0.01%	0.11	0.05%	6.9
split	2	1.50	0.00%	0.10	0.02%	0.25	0.03%	4.3
?	4	0.50	0.00%	0.00	0.00%	0.11	0.03%	4.6
chown	2	1.50	0.00%	0.08	0.01%	0.13	0.02%	7.2
date	3	1.00	0.00%	0.01	0.00%	0.13	0.03%	7.2
q	5	0.60	0.00%	0.01	0.00%	0.07	0.02%	7.5
stty	3	0.67	0.00%	0.01	0.00%	0.11	0.02%	6.0
a	3	0.67	0.00%	0.00	0.00%	0.11	0.02%	6.0
moo	1	14.00	0.02%	0.02	0.00%	0.25	0.02%	52.5
prof	1	2.00	0.00%	0.00	0.00%	0.25	0.02%	8.0
exec	2	1.50	0.00%	0.01	0.00%	0.12	0.01%	12.0
123,,,,,	1	1.00	0.00%	0.00	0.00%	0.18	0.01%	5.5
chdi	1	1.00	0.00%	0.02	0.00%	0.15	0.01%	6.0
sqrt	1	1.00	0.00%	0.00	0.00%	0.17	0.01%	6.0
mount	1	1.00	0.00%	0.02	0.00%	0.15	0.01%	6.0
r	1	1.00	0.00%	0.00	0.00%	0.15	0.01%	6.7
edit	1	1.00	0.00%	0.02	0.00%	0.13	0.01%	6.7
chmod	2	0.50	0.00%	0.00	0.00%	0.08	0.01%	6.7
help	1	1.00	0.00%	0.00	0.00%	0.15	0.01%	6.7
nrof	1	1.00	0.00%	0.02	0.00%	0.12	0.01%	7.5
pwf	1	1.00	0.00%	0.00	0.00%	0.13	0.01%	7.5
core	1	1.00	0.00%	0.00	0.00%	0.13	0.01%	7.5
as2	1	0.00	0.00%	0.00	0.00%	0.13	0.01%	0.0
du	1	1.00	0.00%	0.00	0.00%	0.13	0.01%	7.5
sleep	1	16.00	0.02%	0.00	0.00%	0.13	0.01%	120.0
sqr	1	1.00	0.00%	0.00	0.00%	0.13	0.01%	7.5
nrofile1	1	2.00	0.00%	0.00	0.00%	0.13	0.01%	15.0
cAt	1	1.00	0.00%	0.00	0.00%	0.13	0.01%	7.5
w	1	0.00	0.00%	0.00	0.00%	0.13	0.01%	0.0
logoff	1	1.00	0.00%	0.02	0.00%	0.10	0.01%	8.6
143,144p	1	0.00	0.00%	0.00	0.00%	0.12	0.01%	0.0
fjf	1	0.00	0.00%	0.02	0.00%	0.10	0.01%	0.0
dept_lis	1	0.00	0.00%	0.02	0.00%	0.10	0.01%	0.0
f	1	1.00	0.00%	0.00	0.00%	0.12	0.01%	8.6
l	1	0.00	0.00%	0.00	0.00%	0.10	0.01%	0.0
chlir	1	1.00	0.00%	0.02	0.00%	0.08	0.01%	10.0
phil	1	1.00	0.00%	0.00	0.00%	0.10	0.01%	10.0
8ed	1	1.00	0.00%	0.00	0.00%	0.10	0.01%	10.0

2l	1	1.00	0.00%	0.00	0.00%	0.10	0.01%	10.0
r4	1	1.00	0.00%	0.02	0.00%	0.07	0.00%	12.0
csw	1	1.00	0.00%	0.00	0.00%	0.08	0.01%	12.0
ca	1	1.00	0.00%	0.02	0.00%	0.07	0.00%	12.0
mial	1	1.00	0.00%	0.00	0.00%	0.07	0.00%	15.0
h	1	2.00	0.00%	0.00	0.00%	0.07	0.00%	30.0
cal[jan]	1	0.00	0.00%	0.00	0.00%	0.07	0.00%	0.0

Notes: In this measurement, we do not know what commands produced the "**gok" entries. ("**gok" is a command at the input end of a pipe that is not otherwise identified.) The "?" category includes all commands that consisted exclusively of non-printable characters.

Figure B6: RP03 root-RF11 swap, normal day

Figures B7 through B12 contain data extracted from Figures B1 through B5, and from Figure A1 of Appendix A. These figures are organized to facilitate comparisons between runs. Particularly interesting data are underlined. All times are in seconds.

In Figure B7, the sixth column is the true benchmark command count; in Figure B8 and subsequent figures, the sixth column is data from measurement #6.

In column six of the figures, blank means no commands of this type, and "ID" (Insufficient Data) means fewer than five commands. The "**gok" data for run #6 are also omitted, because they are not comparable to the "**gok" data from Figures B1-B5.

	#1	#2	#3	#4	#5	see text
Total	518	510	506	514	501	524
sh	61	59	59	61	60	60
ed	86	90	82	87	84	90
cc	20	18	19	20	19	20
**gok	10	10	10	10	10	10
colv	10	10	10	10	10	10
pr	20	17	20	19	19	20
ls	59	57	58	56	59	60
cat	50	50	47	47	48	50
du	20	19	20	20	19	20
grep	10	10	10	10	10	10
ps	9	10	9	10	9	10
rm	40	40	39	39	39	40
cp	29	28	30	31	29	30
pwd	10	10	10	10	8	10
mv	21	18	19	21	19	20
rmdir	10	10	10	10	8	10
mkdir	10	10	10	10	8	10
who	9	10	10	10	10	10
echo	10	10	10	10	10	10
date	10	10	10	10	9	10
chmod	10	10	10	9	10	10
sleep	4	4	4	4	4	4
Average	40.81	39.59	37.88	37.55	38.70	77.33
sh	110.00	120.27	116.83	112.41	116.27	36.10
ed	45.73	42.79	43.70	42.86	43.67	230.13
cc	291.65	192.89	177.63	175.35	176.89	33.71
**gok	157.90	158.40	152.60	152.50	162.40	
colv	1.10	1.10	0.80	1.90	1.50	
pr	24.45	32.06	27.95	29.32	26.74	17.38
ls	11.81	13.95	11.59	10.93	11.95	4.52
cat	7.36	13.24	12.89	12.06	12.40	19.89
du	8.00	28.63	25.00	25.05	25.95	ID
grep	13.00	17.40	17.80	15.60	18.30	ID
ps	58.56	48.00	46.11	40.30	45.89	27.83
rm	5.85	8.25	6.77	6.74	7.23	1.23
cp	4.59	9.32	8.50	8.29	8.55	9.77
pwd	6.10	7.80	5.90	6.40	6.75	1.13
mv	3.00	3.78	3.32	3.29	3.11	1.23
rmdir	2.70	3.30	3.60	3.30	4.13	
mkdir	3.40	5.00	4.20	4.70	4.63	
who	2.78	3.60	2.20	2.80	2.60	7.50
echo	1.80	2.00	1.10	1.20	1.30	
date	1.40	1.30	1.50	1.10	1.00	ID
chmod	2.90	2.10	2.00	2.11	1.90	ID
sleep	16.00	17.25	16.75	16.75	16.75	ID

Figures B7 and B8: Command Count; Real Time

	#1	#2	#3	#4	#5	#6
Average	3.05	3.10	3.08	3.07	3.13	1.18
sh	9.78	10.21	10.13	9.89	10.14	0.53
ed	1.51	1.46	1.42	1.40	1.48	1.36
cc	14.70	15.97	15.29	14.92	14.41	11.26
**gok	27.88	28.14	27.63	27.91	28.49	
colv	12.99	13.03	13.00	13.06	13.09	
pr	2.98	3.49	3.02	3.14	2.91	0.06
ls	0.96	1.00	0.96	0.94	0.98	0.07
cat	0.31	0.31	0.33	0.32	0.32	0.01
du	0.07	0.06	0.06	0.06	0.04	ID
grep	1.43	1.39	1.42	1.41	1.39	ID
ps	0.06	0.20	0.26	0.19	0.21	0.20
rm	0.01	0.02	0.02	0.02	0.02	0.00
cp	0.00	0.01	0.01	0.01	0.01	0.03
pwd	0.00	0.01	0.01	0.01	0.01	0.01
mv	0.00	0.01	0.00	0.00	0.01	0.00
rmdir	0.00	0.00	0.00	0.00	0.01	
mkdir	0.00	0.00	0.01	0.00	0.00	
who	0.00	0.01	0.01	0.01	0.01	0.04
echo	0.00	0.01	0.00	0.00	0.01	
date	0.01	0.01	0.00	0.01	0.01	ID
chmod	0.00	0.00	0.00	0.00	0.01	ID
sleep	0.00	0.00	0.00	0.00	0.00	ID
Average	2.81	3.03	2.96	3.03	3.03	1.72
sh	9.49	10.49	10.30	10.31	10.22	4.01
ed	5.43	5.50	5.78	5.66	5.83	1.12
cc	7.05	7.56	7.45	7.30	7.41	4.41
**gok	2.81	2.75	2.82	2.80	2.73	
colv	1.39	1.43	1.40	1.39	1.38	
pr	2.99	2.51	1.45	2.55	1.78	0.37
ls	0.68	0.70	0.70	0.68	0.67	0.21
cat	0.65	0.65	0.65	0.65	0.64	0.14
du	1.20	1.27	1.32	1.22	1.29	ID
grep	0.79	0.77	0.74	0.78	0.79	ID
ps	2.41	6.56	6.50	6.02	6.18	5.44
rm	0.41	0.42	0.44	0.42	0.42	0.16
cp	0.38	0.36	0.37	0.36	0.38	1.76
pwd	0.37	0.36	0.36	0.38	0.37	0.37
mv	0.14	0.13	0.14	0.14	0.11	0.18
rmdir	0.17	0.17	0.15	0.17	0.17	
mkdir	0.13	0.14	0.15	0.13	0.13	
who	0.11	0.12	0.12	0.14	0.13	0.16
echo	0.09	0.10	0.10	0.10	0.11	
date	0.09	0.10	0.10	0.09	0.09	ID
chmod	0.08	0.08	0.08	0.07	0.07	ID
sleep	0.10	0.12	0.08	0.10	0.10	ID

Figures B9 and B10: USER Time; System Time

	#1	#2	#3	#4	#5	#6
Average	5.86	6.13	6.04	6.10	6.16	3.00
sh	19.27	20.70	20.43	20.20	20.36	4.54
ed	6.94	6.96	7.20	7.06	7.31	2.48
cc	21.75	23.53	22.74	22.22	21.82	15.67
**gok	30.69	30.89	30.45	30.71	31.22	
colv	14.38	14.46	14.40	14.45	14.47	
pr	5.97	6.00	4.47	5.69	4.69	0.43
ls	1.64	1.70	1.66	1.62	1.65	0.28
cat	0.96	0.96	0.98	0.97	0.96	0.15
du	1.27	1.33	1.38	1.28	1.33	ID
grep	2.22	2.16	2.16	2.19	2.18	ID
ps	2.47	6.76	6.76	6.21	6.39	5.64
rm	0.42	0.44	0.46	0.44	0.44	0.16
cp	0.38	0.37	0.38	0.37	0.39	1.79
pwd	0.37	0.37	0.37	0.39	0.38	0.38
mv	0.14	0.14	0.14	0.14	0.12	0.18
rmdir	0.17	0.17	0.15	0.17	0.18	
mkdir	0.13	0.14	0.16	0.13	0.13	
who	0.11	0.13	0.13	0.15	0.14	0.20
echo	0.09	0.11	0.10	0.10	0.12	
date	0.10	0.11	0.10	0.10	0.10	ID
chmod	0.08	0.08	0.08	0.07	0.08	ID
sleep	0.10	0.12	0.08	0.10	0.10	ID
Average	7.0	6.5	6.3	6.1	6.3	26.7
sh	5.7	5.8	5.7	5.6	5.7	7.9
ed	6.6	6.1	6.1	6.1	6.0	92.7
cc	13.4	8.2	7.8	7.9	8.1	2.2
**gok	5.1	5.1	5.0	5.0	5.2	
colv	0.1	0.1	0.1	0.1	0.1	
pr	4.1	5.3	6.3	5.2	5.7	40.7
ls	7.2	8.2	7.0	6.7	7.2	15.8
cat	7.7	13.9	13.2	12.4	12.9	131.5
du	6.3	21.6	18.1	19.5	19.5	ID
grep	5.8	8.0	8.3	7.1	8.4	ID
ps	23.6	7.1	6.8	6.5	7.2	4.9
rm	13.8	19.1	14.9	15.6	16.4	7.7
cp	11.8	25.1	22.7	22.2	22.3	5.5
pwd	16.3	21.4	15.9	16.6	17.9	3.0
mv	21.4	27.2	22.6	23.1	27.0	6.5
rmdir	16.0	19.4	23.0	19.6	23.9	
mkdir	26.5	34.9	27.4	37.1	34.2	
who	24.2	27.7	17.4	19.1	18.6	37.5
echo	19.6	19.0	10.5	11.4	10.8	
date	14.0	12.0	14.1	11.4	9.5	ID
chmod	32.8	25.2	22.6	30.8	24.8	ID
sleep	167.0	142.8	223.3	160.8	167.5	ID

Figures B11 and B12: CPU (USER+System) Time; RTR

Figure B13 collects the averages from Figures B8-B12.

	#1	#2	#3	#4	#5	#6
Real Time	40.81	39.59	37.88	37.55	38.70	77.33
USER time	3.05	3.10	3.08	3.07	3.13	1.18
System time	2.81	3.03	2.96	3.03	3.03	1.72
CPU time	5.86	6.13	6.04	6.10	6.16	3.00
RTR	7.0	6.5	6.3	6.1	6.3	26.7

Figure B13: Averages

Figure B14 displays miscellaneous derived data.

	#1	#2	#3	#4	#5	#6
Sum RT	21130	20190	19170	19300	19390	46490
Adj. sum RT	14425	13100	12280	12440	12410	46490
Multiprg level	4.71	4.63	4.55	4.61	4.50	2.94

Figure B14: Miscellaneous Derived Data

The quantities displayed in Figure B14 are defined as follows:

Sum RT — The sum of the real time values for all commands. It is deduced from Figures B1-B6 by multiplying the (mean) "Real Time" for the "ed" command by the number of "ed" commands, and dividing by the "Real Time" percentage. I.e., for measurement #1, "Sum RT" = (45.73)*(86)/(.1861). ("ed" was chosen arbitrarily. Almost any command would have sufficed.)

Adj. sum RT — The special version of the Shell accounting command used to process the data from the benchmark runs causes the time consumed by subcommands of a "sh" command to be counted twice: once under

"sh", and again under the subcommand names. In runs #1-#5, "Adj. sum RT" is "Sum RT" less the fraction of "Sum RT" due to "sh" commands. In run #6, no such adjustment is necessary.

Multiprg level — "Multiprg level" is "Adj. sum RT" divided by measurement duration (from Figure C1 of Appendix C). It is the mean number of commands concurrently in execution on the system.

APPENDIX C: D-7900 Data

	#1	#2	#3	#4	#5	#6
Duration	3060	2830	2700	2700	2760	21420
CPU busy	2124	2146	2114	2133	2701	5865
CPU blocked	35	41	37	40	57	57
CPU wait	900	633	549	527	2	15499
User	943	946	946	950	1527	1684
Kernel	1181	1200	1168	1183	1173	4181
not RUN	1395	1415	1391	1407	1732	3792
PAUSE	1398	1418	1394	1410	1736	3801
Bus MASTER	1667	1682	1659	1673	2100	4587
BBSY	1893	1915	1886	1903	2403	5189
# CPU MSYN	1889M	1910M	1879M	1895M	2303M	5253M
# I/O MSYN	32M	30M	30M	28M	31M	78M
I/O busy	111	98	88	90	58	406
RF11 busy	211	NA	NA	175	209	568
# RF cmds	3412	NA	NA	3193	3373	12320
RP03 busy	972	2361	2178	2242	2192	5014
# RP cmds	38941	86968	83136	84220	83655	167784
RK05 busy	2351	NA	278	NA	NA	328

All times are in seconds. "M" means 1,000,000. "NA" means not applicable.

Figure C1: Raw D-7900 Data

Definitions

In the figures, particularly interesting data are underlined.

Names written in all upper-case letters (e.g. RUN) are used for signals that are defined in Section 8.3 of [1], or in [2]. Names having some lower-case letters (e.g. "Kernel") are used for signals defined and derived by us. Note that these definitions apply only to the PDP-11/45, while running UNIX.

Duration	The duration of the measurement. In the case of the benchmark runs, it is also the duration of the benchmark itself.
CPU busy	The amount of time that the CPU is executing instructions other than wait, not counting the time during which an I/O device is UNIBUS MASTER.
CPU blocked	The amount of time that the CPU is trying to execute instructions other than WAIT, but is blocked because an I/O device is UNIBUS MASTER.
CPU wait	The amount of time spent executing wait instructions (idle). "CPU busy" + "CPU blocked" + "CPU wait" = "Duration".
User	"CPU busy" and USER mode.
Kernel	"CPU busy" and KERNEL mode.
not RUN	The amount of time that the console RUN lamp is off. See text for meaning.
PAUSE	The amount of time that the console PAUSE lamp is on. For a PDP-11/45 with a normal I/O complement, "PAUSE" will always be a fraction of a percent larger than "not RUN".
Bus MASTER	The time-integral of the console MASTER signal. MASTER is a stream of pulses that is present so long as the CPU is UNIBUS MASTER, i.e. busy and using the UNIBUS. Because MASTER is a pulse stream, "Bus MASTER" cannot be as large as 100% of real time.
BBSY	The time-integral of the console BBSY (UNIBUS busy) signal. The BBSY signal is a pulse stream, and thus "BBSY" cannot reach 100% of real time, even when the UNIBUS is truly 100% busy.
# CPU MSYN	A MSYN (Master Sync) pulse occurs on the UNIBUS every time the bus master device places a new address or data word (or both) on the UNIBUS. "# CPU MSYN" is the number of MSYN pulses issued

by the CPU. Almost all of these pulses represent memory references (one pulse per reference).

# CPU MSYN/sec	The mean number of CPU MSYN pulses per second.
# I/O MSYN	The number of MSYN pulses issued by I/O devices.
# I/O MSYN/sec	The mean number of I/O MSYN pulses per second.
I/O busy	The amount of time during which I/O devices are using the UNIBUS. If the CPU and an I/O device are making interleaved use of the UNIBUS, only the time consumed by I/O data transfers is counted in "I/O busy".
RF11 busy	The amount of time that the Control Ready (RDY) bit in the Disk Control Status Register of the RF11 (RS11 controller) is not set. "RF11 busy" includes both rotational latency time and data transfer time.
# RF cmds	The number of "RF11 busy" signals. There is one signal for each RF11 command.
# RF/sec	The mean number of RF11 commands per second.
RP03 busy	The amount of time that the READY bit in the Control Status Register of the RP11 (RP03 controller) is not set. "RP03 busy" includes rotational latency time, data transfer time and write check time, and, because of the way the RP03 driver is programmed, seek time.
# RP cmds	The number of "RP03 busy" signals. There is one signal for each seek command, data transfer command or write check command.
# RP/sec	The mean number of RP03 commands per second.
RK05 busy	The amount of time that the Control Ready (RDY) bit in the Control Status Register of the RK11 controller is not set. "RK05 busy" includes both rotational latency time and data transfer time.

Measuring CPU Behavior

So far as the CPU is concerned, real time splits into three parts: "CPU busy", "CPU blocked" and "CPU WAIT". (See above for definitions.) Unfortunately, we have been unable to find a simple way to measure these times directly on the PDP-11/45. Instead, we have used the following indirect method.

	#1	#2	#3	#4	#5	#6
CPU busy	69.42 100.00	76.10 100.00	78.29 100.00	79.00 100.00	97.86 100.00	27.38 100.00
CPU blocked	1.16 1.67	1.45 1.91	1.36 1.74	1.47 1.86	2.07 2.12	0.26 0.95
CPU wait	29.43 42.39	22.46 29.51	20.35 25.99	19.53 24.72	0.07 0.07	72.36 264.28
User	30.82 44.40	33.54 44.08	35.03 44.74	35.19 44.54	55.34 56.55	7.86 28.71
Kernel	38.59 55.59	42.55 55.91	43.25 55.24	43.80 55.44	42.50 43.43	19.52 71.29
not RUN	45.60 65.69	50.18 65.94	51.52 65.81	52.12 65.97	62.76 64.13	17.71 64.68
PAUSE	45.69 65.82	50.29 66.08	51.64 65.96	52.24 66.13	62.91 64.29	17.75 64.83

All times are given twice: first as % of "Duration", then as % of "CPU busy".

Figure C2: Normalized D-7900 CPU Data

There is a signal line in the UNIBUS called MSYN (Master Sync). Every word of data (which may be one or two bytes) that is transmitted over the UNIBUS, including every core memory reference, is accompanied by one MSYN pulse. So long as the system is busy, there is a steady stream of MSYN pulses, at a rate of about one every microsecond.

It is possible to distinguish between CPU-generated and I/O-generated MSYN pulses. If the console MASTER signal is true when a MSYN pulse is detected, then the MSYN pulse originated in the CPU. If the MASTER signal is false, then the MSYN pulse originated in an I/O device.

We have wired the D-7900 plugboard to monitor the stream of MSYN pulses. So long as there is an uninterrupted stream of CPU MSYN pulses, time is credited to "CPU busy". If one or more I/O MSYN pulses immediately follow a CPU MSYN pulse, then we assume that the CPU was preempted, and credit the time to "CPU blocked" and to "I/O busy".

If there is a gap of at least 5 microseconds in the MSYN pulse stream, we assume that the CPU is executing a WAIT, and count all time until the next CPU MSYN pulse as "CPU WAIT". If any I/O MSYN pulses occur before the next CPU MSYN pulse, their time is credited to "I/O busy" (but not to "CPU blocked").

	#1	#2	#3	#4	#5	#6
Bus MASTER	54.49 78.49	59.64 78.37	61.46 78.50	61.97 78.44	76.07 77.74	21.42 78.23
BBSY	61.87 89.12	67.92 89.25	69.87 89.25	70.47 89.20	87.07 88.98	24.23 88.49
# CPU MSYN/sec	.6174M .8894M	.6774M .8901M	.6960M .8890M	.7020M .8886M	.8345M .8550M	.2452M .8955M
# I/O MSYN/sec	10297 14833	10820 14218	10998 14048	10724 13574	11102 11349	3621 13225
I/O busy	3.62 5.20	3.47 4.56	3.24 4.14	3.32 4.21	2.09 2.14	1.89 6.90
RF11 busy	6.91 9.95	NA NA	NA NA	6.47 8.19	7.56 7.72	2.65 9.68
# RF/sec	1.12 1.61	NA NA	NA NA	1.18 1.50	1.22 1.25	0.57 2.09
RP03 busy	31.77 45.77	83.73 110.03	80.65 103.01	83.03 105.10	79.41 81.15	23.41 85.50
# RP/sec	12.73 18.33	30.84 40.53	30.79 39.33	31.19 39.48	30.31 30.97	7.83 28.60
RK05 busy	76.82 110.66	NA NA	10.29 13.14	NA NA	NA NA	1.53 5.59

All times are given twice: first as % of "Duration", then as % of "CPU busy". Similarly, event rates are given twice: as events per second of "Duration", and as events per second of "CPU busy". "M" means 1,000,000. "NA" means not applicable.

Figure C3: Normalized D-7900 UNIBUS and I/O Data

There is a console lamp called "RUN". This lamp is on while the CPU is executing microcycles, off while the CPU is waiting for data from memory or from an I/O device, and off while waiting for a DMA device to relinquish the UNIBUS. This means that while executing a WAIT instruction the RUN lamp is on; while executing other instructions the RUN lamp of an 11/45 system with core memory is off most of the time, because the CPU is much faster than the core memory.

Because the "RUN" lamp is on for the full duration of all WAIT instructions, and for part of the duration of all other instructions, "not RUN" is strictly less than, but roughly proportional to, "CPU busy". See the "Analysis of D-7900 Data" section of this memorandum for an estimate of the constant of proportionality.

APPENDIX D: D-8000 Data

Routine	#1	#2	#3	#4	#5	#6
_access	0.08	0.05	0.05	0.05	0.03	0.02
_alloc	0.03		0.03	0.03		
_aretu, _retu	0.57	0.57	0.59	0.57	0.63	0.38
_backup						
_badbloc	0.05	0.08	0.08	0.05	0.07	
_bawrite				0.03		
_bcopy	0.65	0.44	0.27	0.52	0.40	0.66
_bdwrite	0.16	0.18	0.19	0.18	0.20	0.19
_bflush						
_binit						
_bmap	1.89	1.95	2.03	1.92	1.94	0.64
_bread	1.01	0.95	1.04	0.98	1.03	0.34
_breada	0.21	0.21	0.21	0.21	0.17	0.11
_brelse	2.18	2.19	2.27	2.20	2.27	0.76
_bwrite	0.03	0.03	0.03	0.03	0.03	0.02
_canon						0.13
_chdir						
_chmod						
_chown						
_cinit						
_clearse	0.88	0.85	0.88	0.85	0.90	0.63
_clock	0.86	0.82	0.83	0.80	0.90	1.80
_close						
_closef						
_closei						
_clrbuf	0.86	0.85	0.88	0.85	0.87	0.27
_copyin, _copyout	2.57	2.55	2.67	2.56	2.67	2.14
_copyseg	2.00	2.03	2.11	2.05	2.04	1.08
_core						
_cpass	0.26	0.21	0.13	0.21	0.17	0.28
_creat						
_devstar	0.39	0.39	0.40	0.39	0.43	0.23
_dhclose						
_dhopen						
_dhparam						0.04
_dhread						0.09
_dhrint						
_dhsgtty						1.53
_dhstart						0.04
_dhwrite						3.16
_dhxint						
_display	0.44	0.39	0.40	0.39	0.37	0.83
_dmint						
_dmopen						
_dpadd	0.18	0.21	0.21	0.21	0.20	0.11
_dpcmp	0.60	0.64	0.67	0.65	0.67	0.28

_dup						
_estabur	0.03	0.03	0.03	0.03		0.02
_etext						
_exec	0.08	0.10	0.11	0.10	0.07	0.08
_exit	0.16	0.15	0.16	0.16	0.17	0.13
_expand	0.10	0.13	0.11	0.10	0.13	0.08
_falloc	0.03		0.03	0.03	0.03	0.04
_flushtt						
_fork						
_free		0.03		0.03		
_fstat						
_fubyte	0.62	0.59	0.40	0.57	0.50	0.57
_fuword	3.53	3.55	3.49	3.55	3.61	2.69
_getblk	5.48	8.26	5.89	8.11	8.39	2.33
_getc						1.08
_geterro	0.08	0.05	0.08	0.05	0.03	0.06
_getf	0.52	0.54	0.56	0.52	0.57	0.32
_getfs	0.10	0.10	0.11	0.10	0.10	0.06
_getgid						
_getmdev						
_getpid						
_getswit						
_getuid						
_gtime						
_gtty						
_ialloc						
_idle	0.08	0.05	0.05	0.05		0.27
_ifree						
_iget	1.79	1.47	1.52	1.35	1.57	1.12
_iinit						
_incore	0.60	0.72	0.75	0.72	0.77	0.30
_incupc						
_iodone	0.13	0.13	0.13	0.13	0.17	0.09
_iomove	1.56	1.41	1.31	1.45	1.44	0.78
_iowait	0.31	0.31	0.35	0.31	0.33	0.17
_iput	0.05	0.05	0.05	0.18	0.03	0.02
_issig	0.36	0.39	0.40	0.39	0.37	0.27
_itrunc	0.10	0.10	0.11	0.10	0.10	0.02
_iupdat	0.05	0.03	0.03	0.03	0.03	0.13
_kill						
_klclose						
_klopen						
_klread						
_klrint						
_klsgtty						
_klwrite						
_klxint						
_ldiv, _lrem	0.08	0.13	0.11	0.10	0.10	0.08
_link						0.47
_lpcanon						
_lpclose						0.06
_lpint						
_lpopen						

_lpoutpu						0.23
_lpstart						0.32
_lpwrite						0.09
_lrem						
_lshift	0.18	0.26	0.27	0.23	0.27	0.09
_main						
_maknode						
_malloc						
_max			0.03			0.04
_mfree						0.04
_min	0.70	0.69	0.72	0.70	0.74	0.28
_mknod						
_mmread	0.05	0.46	0.40	0.34	0.43	
_mmwrite						
_namei	1.61	1.44	1.52	1.50	1.47	0.93
_newproc	0.10	0.10	0.11	0.10	0.07	0.08
_nice						
_nodev,	0.26	0.18	0.11	0.21	0.13	0.25
_nulldev						
_nosys	0.36	0.39	0.43	0.39	0.40	0.23
_notavai	1.45	1.49	1.55	1.48	1.54	0.51
_nseg						
_nulldev	Combined with _nodev					
_nullsys						
_open						
_open1						
_openi						
_owner						
_panic						
_passc	0.86	1.10	1.07	1.01	1.10	0.43
_pcclose						
_pcleade						
_pcopen						
_pcoutpu						
_pcpint						
_pcread						
_pcrint						
_pcstart						
_pcwrite						
_physio						
_pipe						0.06
_plock						
_prdev						
_prele	0.05	0.03	0.05	0.03	0.03	0.15
_printf						
_printn						
_profil						
_psig						
_psignal						0.93
_putc						
_putchar	0.03					
_rdwr	1.56	1.57	1.65	1.61	1.64	0.83
_read	0.21	0.21	0.21	0.21	0.23	0.06
_readi	3.25	3.24	3.41	3.29	3.31	0.68

_readp						
_retu	Combined with _aretu					
_rexit						
_rfintr						
_rfread						
_rfstart						
_rfstrat						
_rfwrite						
_rkaddr	0.29					
_rkintr	0.18					
_rkread						
_rkstart	0.16					
_rkstrat	0.29					
_rkwrite						
_rpintr	0.18	0.36	0.37	0.36	0.33	0.19
_rpphys						
_rpread						
_rpstart	0.16	0.31	0.32	0.31	0.30	0.19
_rpstrat	0.49	1.44	1.41	1.32	1.37	0.51
_rpwrite						
_savfp	0.70	0.72	0.75	0.72	0.80	0.47
_savu	0.39	0.36	0.40	0.36	0.40	0.28
_sbreak						
_schar						
_sched	0.16	0.15	0.13	0.13	0.13	0.13
_seek	0.08	0.08	0.08	0.08	0.10	0.06
_setgid						
_setrun	0.13	0.10	0.11	0.10	0.13	0.11
_setuid						
_sgtty						
_signal						
_sleep	0.18	0.18	0.19	0.18	0.20	0.19
_smdate						
_smount						
_snstat						
_spl0, _spl1, _spl2, _spl3, _spl4, _spl5, _spl6, _spl7	0.44	0.44	0.45	0.44	0.47	0.30
_ssig						
_sslep						
_stat						
_stat1	0.13	0.13	0.13	0.13	0.10	0.04
_stime						
_stiucha						
_stty						
_subyte	0.88	1.05	1.01	1.01	1.04	0.51
_sumount						
_sureg	2.60	2.52	2.64	2.62	3.07	1.69
_suser						
_suword	2.80	2.93	2.93	2.90	2.97	1.97
_swap						
_swtch	19.70	17.61	17.84	17.07	14.37	35.13
_sync						
_tcclose						
_tcintr						

_tcomman						
_tcstart						
_tcstrat						
_timeout						0.02
_times						
_tiubusy						
_tiuchan						
_tiuchec						
_tiucher						
_tiuclos						
_tiuerr						
_tiuintr						
_tiuopen						
_tiuptr						
_tiuread						
_tiustar						
_tiustop						
_tiustra						
_tiuwrit						
_tmclose						
_tmintr						
_tmopen						
_tmphys						
_tmread						
_tmstart						
_tmstrat						
_tmwrite						
_toyfl						
_trap	4.96	5.07	5.25	5.13	5.25	2.92
_trap1						
_ttread						0.06
_ttrstrt						
_ttstart						0.09
_ttwrite						0.40
_ttyinpu						0.09
_ttyoutp						1.17
_tyaintr						
_tybintr						
_tyopen						
_tyread						
_tysgtty						
_tywrite						
_uchar	0.10	0.10	0.11	0.10	0.10	0.06
_ufalloc		0.03				
_unlink						
_update						0.04
_wait						
_wakeup	3.69	3.47	3.57	3.47	3.58	4.36
_wdir						
_wflusht						
_write	0.03	0.03	0.03	0.03	0.03	0.08
_writei	0.39	0.46	0.48	0.47	0.50	0.95
_writep						0.19
_wstiuch						

_xallloc						
_xccdec						
_xfree						
_xswap	0.86	0.69	0.75	0.72	0.80	0.64
call	3.40	3.34	3.47	3.34	3.61	3.35
csv,cret	14.25	13.89	14.35	14.04	14.47	10.57

All data are expressed as the percentage of total time spent in UNIX text segments. A blank indicates that the time spent in the routine was less than 0.005%.

Figure D1: Breakdown of UNIX Time by Routine

Functions of the Most Time-consuming Routines (Fig. D2)

call — Call is the trap and interrupt handling interface to the low memory vectors.

_clock — _clock is the clock interrupt handler. Some of its functions are: to keep track of the software clock, to keep track of user and system time, and to keep track of execution time.

_copyin — _copyin copies a specified number of bytes from user address space to kernel space. _copyout copies a specified number of bytes from kernel space to user space. This is done in conjunction with reading and writing a file. The data are moved back and forth between system buffers before reading and writing.

_copyout — See _copyin.

_copyseg — _copyseg copies a 32 word segment from one area to another area. Forks, expansions of a process, and stack violations which result in the creation or growth of a process are first attempted in memory. _copyseg is used for the in-memory copy.

cret,csv — Cret and csv are routines used by every call generated by the C compiler to save (csv) and restore (cret) registers.

_dhstart — _dhstart places a teletype output character on the proper line of the DH multiplexer.

_dhxint — _dhxint is the DH multiplexer transmit interrupt handler.

Routine	#1	#2	#3	#4	#5	#6
_swtch	19.70	17.61	17.84	17.07	14.37	35.13
csv,cret	14.25	13.89	14.35	14.04	14.47	10.57
_wakeup	3.69	3.47	3.57	3.47	3.58	4.36
call	3.40	3.34	3.47	3.34	3.61	3.35
_dhxint						3.16
_trap	4.96	5.07	5.25	5.13	5.25	2.92
_fuword	3.53	3.55	3.49	3.55	3.61	2.69
_getblk	5.48	8.26	5.89	8.11	8.39	2.33
_copyin, _copyout	2.57	2.55	2.67	2.56	2.67	2.14
_suword	2.80	2.93	2.93	2.90	2.97	1.97
_clock	0.86	0.82	0.83	0.80	0.90	1.80
_dhstart						1.53
_sureg	2.60	2.52	2.64	2.62	3.07	1.69
_ttyoutp						1.17
_iget	1.79	1.47	1.52	1.35	1.57	1.12
_copyseg	2.00	2.03	2.11	2.05	2.04	1.08
_getc						1.08
_writei	0.39	0.46	.048	0.47	0.50	0.95
_namei	1.61	1.44	1.52	1.50	1.47	0.93
_putc						0.93
_display	0.44	0.39	0.40	0.39	0.37	0.83
_rdwr	1.56	1.57	1.65	1.61	1.64	0.83
Total	71.63	71.37	70.61	70.96	70.48	82.56

The twenty-two most time-consuming routines in run #6, sorted in decreasing order. Data from the benchmark runs for the same routines are also included. Particularly interesting data are underlined.

Figure D2: Most Time-Consuming Routines

_display _display loads the console lights with the word addressed by the console switches.

_fuword _fuword gets a word from the user's virtual address space. It is used to get requests for system calls, overlays, etc., from the user's address space.

_getblk _getblk accesses a block on a logical device. It is used by the I/O system to read and write blocks to and from files.

_getc — _getc gets a character from a character I/O buffer and also does deallocation of character I/O buffers.

_iget — _iget returns a pointer to an inode. If the requested inode is not in memory, a copy of the inode is brought into memory from the file system and placed in the inode table.

_namei — For a given pathname, _namei finds and brings into memory the inode for the desired file.

_putc — _putc puts a character on the character output device queue and also does allocation of character I/O buffers.

_rdwr — _rdwr is the interface for all user system calls which do reads or writes (getc, putc, putchar, getchar, read, write, printf, getw, putw, etc.). It determines whether a pipe or a file (including special files) is to be read or written and calls the appropriate system functions to fulfill the request.

_sureg — _sureg relocates a user's memory management registers to the proper place in memory. It is called as part of switching from one process to another.

_suword — _suword is used to store a word in the user's virtual address space. It is most commonly used to initialize the stack in an overlaid process.

_swtch — _swtch does a process switch to the highest priority process in memory that is ready to execute. Round robin scheduling is done with processes of equal priority.

_trap — _trap handles processor traps from user mode (only).

_ttyoutp — _ttyoutp puts a character on the TTY output queue, calculates the required delay, performs upper-case translation if required by the terminal, etc.

_wakeup — _wakeup wakes up processes when specific events occur.

_writei — _writei writes the specified number of bytes to a file or device. It provides write capability to special files (block and character devices) and to UNIX file systems.

APPENDIX E: Measurement Conditions

For convenience when reading Appendices A-D, Figure 3 is repeated here. See the main body of the memorandum for discussion.

	#1	#2	#3	#4	#5	#6
Load	------Benchmark-------				*	Normal
Root	RK05	RP03	RP03	RP03	RP03	RP03
Swap	RF11	RP03	RK05	RF11	RF11	RF11
/usr	----------RP03----------					

*Benchmark plus "cycle sponge". See text.

Figure E1: Measurement Conditions