Banksys

LAN PC Engineering

# IO TEST

## PC Disk IO Analyze and Simulation Program

*This is the abstract*

Ref: PCDOC-000-0

**November 1993**

**Revision/Update Information:** This is a new document

# Contents


Overview . . . . . 1-1
Distribution . . . . . 1-1
How it was written . . . . . 1-2

IO_SNOOP . . . . . 2-1
Parameters description . . . . . 2-1

IO_LIST . . . . . 3-1
Parameters description . . . . . 3-1
"Summary" display description . . . . . 3-6
Analyse Strategy . . . . . 3-7

IO_SIMUL . . . . . 4-1
Parameters description . . . . . 4-2
Output description . . . . . 4-3

Appendix . . . . . 5-1
Miscellaneous programs . . . . . 5-1
IO_SNOOP Dump File Format . . . . . 5-2

# 1

# Overview

This suite of programs was designed with two goals in mind. We first needed a tool to **accurately simulate and time** the disk Input/Output (IO) load of a PC. This program can monitor the variations in response time on a network for a given period. It will enable the network administrator to have scientific clues to judge the typical user complaint: "The network is dreadfully slow this morning!". Another use of the simulator is to accurately represent a PC IO load in a sizing benchmark. It will help to answer questions like "What is the maximum number of workstations of a particular type that a given server can handle?".

The second goal was to help us to **understand the IO structure** of a given PC usage. This tool should assist us in determining what files are opened by an application, what are the "HotFiles" (most often used files) and thus worthwhile to move to a faster media.

The first program of the suite is **IO_SNOOP**, the IO monitoring program. It is loaded as a TSR and will produce a dump of all disk Input/Output operations of a PC. **IO_LIST** is necessary to interpret the binary dump file. It can produce a summary, a detailed list of every IO (one line per IO) or a listing of "HotFiles" . Its primary purpose is to generate a list of parameters to be used as an input for the simulation program: **IO_SIMUL**.

## Distribution

This software is a FreeWare also called BeerWare ("Buy me a beer if you liked it"). Usage and distribution is free. Money may only be charged to cover distribution costs. IO_TEST can be used without restrictions even in a professional or consulting environment.

Comments, bug reports or suggestions for improvements are welcomed. If it was really of great use to you and you want to make

me a pleasure, you can either buy me a beer if we meet, buy me good computer books as they tend to be fairly expensive here in Belgium or buy some toys for my one year old daughter, Deborah, so that she stops pushing on the reset button of my PC ;-)

The following files constitute the distribution kit:

**IOTEST.ZIP**

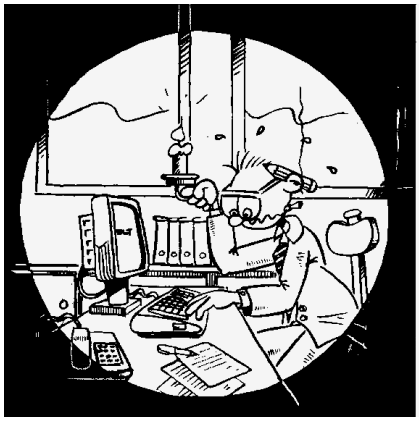
| | |
|---|---|
| IO_SNOOP.EXE | Monitoring TSR |
| IO_LIST.EXE | Dump Analyse Program |
| IO_SIMUL.EXE | PC Load simulator |
| IO_DUMP.EXE | Dump file debugging tool |
| STRDATE.EXE | Sets date as environment variable |
| MERGEDBF.EXE | Concatenates HotFiles databases |
| PAK.BAT | Sample self-extracting test distribution kit building procedure |
| INSTALL.BAT | Sample test distribution kit installation procedure |
| IO_TEST.BAT | Sample test procedure |
| IO_TEST.TXT | ASCII documentation |

**IODOC.ZIP**

| | |
|---|---|
| IO_TEST.PS | Postscript documentation |

# How it was written

IO_TEST was written as an activity of the "*Banksys Midnight Project Computer Club*". This informal club handles PC programming projects that cannot be justified during normal working hours but are useful for the Company. Practical exploration of all aspects of the computer technology is the primary drive of our club. Its activities are sponsored by the "Lan & PC Engineering" group of Banksys (whose member form the core of the club). Banksys is the company handling the debit and credit card network in Belgium.



After Midnight, power is turned off....

The programs were written with the Borland C++ compiler version 3.1. Some special functions were implemented by using specialized libraries as "*CodeBase 4.5*" for the database part, "*Resident C*" for the TSR and "*UseLib*" for the PathWork specific informations.

IO_TEST has been mainly written by myself but with the great help of Michel Valentin who wrote IO_SNOOP. Without his help this project wouldn't have turned out like it has. Many thanks to Chris Lord from Digital Central Engineering (PCI group) who gave me many encouragements and very good advice in numerous informal projects. I would like to take this opportunity to thank Chris for his top notch (and spectacular) presentations during the DECUS Symposiums (Digital Users Society) and especially his "Advanced Topics in PC Memory Management" seminar. Many thanks to Alan Sharkey of Digital (UK) for his fruitful input in discussions in the early stages of the project. I would also like to extend my thanks to Scott Barrielle and Brad Cooper of ISA Consulting for their encouragement.

A very special thank to my wife Ghislaine, for her tender support and quiet patience for my (sometimes excessive) enthusiasm and long programming hours at home.

Brussels, December1993,

Jean-Marc MEESSEN
Banksys LAN & PC Engineering Coordinator

| | |
|---|---|
| Compuserve: | 100016,661 |
| Internet: | 100016.661@COMPUSERVE.COM |
| Phone: | (32)-2-727-6004 |
| Fax: | (32)-2-256-6438 |
| Mail: | Banksys sa/nv<br>Chaussée de Haecht, 1442<br>B-1130 Bruxelles<br>BELGIUM |

# 2

# IO_SNOOP

This program is the TSR that will collect all disk IO informations and dump them to a binary file. The DOS interrupt 21h is hooked so that the disk IO functions can be monitored. Disk IO under Windows can be monitored as long as they are done through DOS int 21h.

It takes approximatively 22.8 Kb of conventional memory. IO_SNOOP has mainly been tested under MS-DOS 5.0. Some tests have been conducted under V6.0.

Usage :

```
IO_SNOOP pathname | /U | /D | /E
```

| | |
|---|---|
| pathname : | output filename. |
| /U : | Unload IO_SNOOP. |
| /D : | Disable logging. |
| /E : | Enable logging. |

## Parameters description

**pathname**: This is the name of the binary dump file that will contains the IO informations. The data is stored in a 4 Kb buffer that is flushed to the dump file when it gets full. This must be kept in mind if the computer is turned off without unloading IO_SNOOP or disabling logging.
If a file of that name already exist, the new information is appended to it.

**/U:** Allows to unload the program from memory if it was the last to be loaded. The buffers are flushed and the dump file closed.

**/D:** Disable the logging of IO events and flushes the buffers and closes the dump file.

**/E:** Re-enables the logging after it has been suspended by /D.

The following INT 21h Functions logged:

Note: a full description of the binary dump file layout is available at the end of this documentation.

0F : FCB open
10 : FCB close
11 : FCB search first
12 : FCB search next
13 : FCB delete
14 : FCB read
15 : FCB write
16 : FCB create
21 : FCB random read
22 : FCB random write
27 : FCB random block read
28 : FCB random block write
3C : Create file
3D : Open file
3E : Close file
3F : Read file
40 : Write file
41 : Delete file
42 : Lseek
4B : Exec file
4E : Find first
4F : Find next
5C : Flock
6C : Extended open/create file

# 3

# IO_LIST

This program will analyse a dump file produced by IO_SNOOP and produce different types of outputs: a summary, a detailed log, a "HotFiles" listing or a profile that can be used as an input for the simulation program.

Usage:

```
IO_LIST input filename [switches] [@response file]
```

| | |
|---|---|
| Input_Filename | Name of the IO_SNOOP dump file to be analysed |
| @file | specifies a response file name (list of commands) |
| /? | Prints this help text |
| /Output[=file] | Specifies an output file |
| /Dbase_output[=file] | Specifies a DBASE output file |
| /Spreadsheet[=file] | specifies a file for a spreadsheet usable summary |
| /Simulator[=file] | generates an input file for the IO Simulator Program |
| /Full | full detail (one line per IO) |
| /Hotfiles | analyses the access on a "per file" base |

Note: commands are not case sensitive and can be abbreviated up to the point where it becomes ambiguous.

## Parameters description

**Input Filename** indicates the path of the dump file generated by IO_SNOOP.

**@file** will execute the list of valid IO_LIST command contained in the file. It can contain a single or multiple commands per line. A line starting with ";" is considered as a command.

**/?** Displays a list of valid commands.

**/Output[=file]** The analyse and/or the full display is redirected to the specified file, or if the name is omitted, to the input filename with the suffix "**.OUT**". If the file already exists, the user has the choice between creating a new one, appending to the existing or quitting the program.
If no output is specified, all the requested informations are displayed to the screen. At every screenfull, the display is paused and the user can choose either to show the next screenfull, show it without pausing or aborting.

A typical summary output can be seen here after. For a more detailed explanation, see the section "Summary display description".

```
Capture start at 11-NOV-1993  20:45:06

Summary:

   Input file: TEST.DMP

   Start time: 20:45:06
   End Time:   23:52:58

   Elapsed Time: 3 hour, 7 min, 52 sec   (11272 sec)

        FCB SearchFirst        13
        FCB SearchNext        146
        FCB Delete              1
        Create File           109
        Open File             552
        Real Close File       659
        Dummy Close File      445
        Delete File            15
        Lseek                1417
        Execute file           17
        Find First           1714
        Find Next             375
        Seq Large Read       1912     Average size:   4107 Bytes
        Rnd Large Read        590     Average size:   4389 Bytes
        Seq Small Read       4961     Average size:   1900 Bytes
        Rnd Small Read        427     Average size:   1396 Bytes
        Seq Large Write       673     Average size:   4766 Bytes
        Rnd Large Write       198     Average size:   8005 Bytes
        Seq Small Write       780     Average size:   1461 Bytes
        Rnd Small Write        30     Average size:    721 Bytes

        TOTAL             15035

List of executed files
    20:45:37.8   C:\BORLANDC\BIN\BC.EXE
    20:45:41.7   C:\BORLANDC\BIN\DPMILOAD.EXE
    20:48:37.9   C:\COMMAND.COM
    22:13:05.0   C:\COMMAND.COM
    22:13:09.6   C:\JMM\IO_SNOOP\IO_SNOOP.EXE
(...)

Small Read and Write repartition            (Size smaller then 4000 bytes)

            Seq Read   Rnd Read  Seq Write   Rnd Write
<   100        1054         71        232          20
<   200          41         28          0           0
<   300          72          6          0           0
<   400          40         22          0           0
<   500          38         13          0           0
<   600          76          9          1           0
(...)

Large Read and Write repartition            (Size larger then 4000 bytes)

            Seq Read   Rnd Read  Seq Write   Rnd Write
<  5500        1911        575        657         180
<  7000           0          6          0           0
<  8500           0          3          0           0
< 10000           0          0          0           0
< 11500           0          0          0           0
< 13000           0          0          0           0
(...)
```

**/Dbase_output[=file]**

Specifies whether a detailed interpretation of the dump file is stored in an Xbase format (one record per IO). If the name is omitted, it will default to the input filename with the suffix "**.DBF**". In the case that the file already exists, the user has the choice between creating a new one, appending to the existing or quitting the program.

The database layout is:

| Field | Type | Width | Decimal |
|---|---|---|---|
| TIME | C | 12 | 0 |
| FUNCTION | C | 2 | 0 |
| HANDLE | N | 2 | 0 |
| F_NAME | C | 80 | 0 |
| SIZE | N | 10 | 0 |

**/Spreadsheet[=file]**

This command will generate a summary in an ASCII file format, delimited so that it can be imported in a spreadsheet. It has been tested with Lotus 123 for Windows (v1.1). If the name is omitted, it will default to the input filename with the suffix "**.PRN**". In the case that the file already exists, the user has the choice between creating a new one, appending to the existing or quitting the program.

**/Simulator[=file]**

With this command, a simulator input data file will created. If the name is omitted, it will default to the input filename with the suffix "**.SIM**". In the case that the file already exists, the user has the choice between creating a new one, appending to the existing or quitting the program.
This file will be a synthesis of all IO collected so that the suggested values for the simulator will total up to 2000 IO per test.

```
; IO_SIMUL input file (usage: "IO_SIMUL [options] @TEST.SIM)"
; Ratio used: 7
;
;
BigReadCnt         =  273   BigReadSize         =   4107
SmallReadCnt       =  708   SmallReadSize       =   1900
BigWriteCnt        =   96   BigWriteSize        =   4766
SmallWriteCnt      =  111   SmallWriteSize      =   1461
RandBigReadCnt     =   84   RandBigReadSize     =   4389
RandSmallReadCnt   =   61   RandSmallReadSize   =   1396
RandBigWriteCnt    =   28   RandBigWriteSize    =   8005
RandSmallWriteCnt  =    4   RandSmallWriteSize  =    721
Create             =   15
OpenClose          =   78
FindFirst          =  244
FindNext           =   53
```

**/Full** This toggle switch allows to display (or save to a file defined by /OUTPUT) one line per Input/Output operation.

This is a sample output.

```
Capture start at 11-NOV-1993  14:45:06
14:45:06.0 <- CloseFile      (#05) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#06) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#07) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#08) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#09) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#10) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#11) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#12) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#13) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#14) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#15) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#16) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#17) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#18) empty SFT - Dummy close
14:45:06.0 <- CloseFile      (#19) empty SFT - Dummy close
14:45:37.2   FindFirst       bc.???
14:45:37.2   FindNext
14:45:37.2   FindFirst       C:\DOS\bc.???
14:45:37.3   FindFirst       C:\WINDOWS\bc.???
14:45:37.4   FindNext
14:45:37.4   FindNext
14:45:37.4   FindFirst       C:\MOUSE\bc.???
14:45:37.5   FindFirst       C:\UTIL\bc.???
14:45:37.6   FindFirst       C:\GMKW\bc.???
14:45:37.7   FindFirst       C:\BORLANDC\BIN\bc.???
14:45:37.7   FindNext
14:45:37.7   FindNext
14:45:37.7   FindNext
14:45:37.8   ExecuteFile     C:\BORLANDC\BIN\BC.EXE
14:45:38.0 -> OpenFile       (#5/3) C:\BORLANDC\BIN\BC.EXE
14:45:38.0   LseekFile       (#5/3) C:\BORLANDC\BIN\BC.EXE
14:45:38.0   LseekFile       (#5/3) C:\BORLANDC\BIN\BC.EXE
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
14:45:38.1   ReadFile        (#5/3) C:\BORLANDC\BIN\BC.EXE     3072 Bytes
(...)
```

The first line of this sample indicates when the measurement started. There will be such a line every time the capture is started or re-started after a suspension. The IO time is given in "Hour:Minute:Second.TenthOfSeconds" format. The next element is a mnemonic word representing the IO operation. It is prefixed with a "->" for an open or assimilated operation and a "<-" for a close operation. The group `(#x/y)` is composed of the handle number (x) and the SFT (System File Table) entry number (y) corresponding to the handle.

The sample where the Borland IDE was started just after the capture, can interpreted in the following way. Exiting IO_SNOOP causes DOS to close automatically all file handles even if they have not been assigned. This type of IO is called here a "dummy close". The command "BC" is then typed at the prompt. This causes the path to be searched to find the BC.EXE file. Half a second and 13 IOs were necessary to find it. This shows the necessity to keep the search path as short as possible! As the file has been found, it is then executed, that means opened and read in 3072 bytes chunks.

**/Hotfiles** This other toggle switch will generate a list of accessed files sorted in descending order. It will be displayed either on screen or written into the file designated by the "/Output" parameter. To build the list, a temporary database file is generated (HOTFILES.DBF) as its associated FoxPro Index file HOTFILES.CDX.

```
Filename                                Open        Read       Write       Total

C:\BORLANDC\BIN\BC.EXE                     6        3533           0        3539
C:\JMM\IO_SNOOP\IO_LIST.SYM               38         662         508        1208
C:\JMM\IO_SNOOP\IO_LIST.EXE               18         486         336         840
C:\BORLANDC\LIB\CL.LIB                    16         688           0         704
C:\JMM\IO_SNOOP\B4.LIB                    16         574           0         590
C:\JMM\BENCHDSK\IO_SIMUL.SYM              16         289         204         509
C:\JMM\IO_SNOOP\IO_LIST.C                 60          23         320         403
C:\JMM\IO_SNOOP\LLIST.OBJ                  8         258          12         278
C:\JMM\IO_SNOOP\IO_LIST.OBJ                7         140          80         227
C:\JMM\BENCHDSK\IO_SIMUL.C                23          27          90         140
C:\BORLANDC\LIB\MATHL.LIB                 16         103           0         119
C:\COMMAND.COM                            13          98           0         111
C:\BORLANDC\BIN\TCCONFIG.TC               12          69           0          81
C:\BORLANDC\LIB\EMU.LIB                   16          64           0          80
C:\BORLANDC\BIN\DPMI16BI.OVL               9          63           0          72
C:\JMM\IO_SNOOP\PARSECMD.OBJ               7          56           8          71
```

FILENAME is the full file name of the file accessed. OPEN, READ and WRITE is the type of access made to the file. The TOTAL column is the sum of the three categories of IO. It's on that information that the sort is done.

Different "HotFiles" analyse database can be merged and sorted using the same method by using the MERGEDBF utility (see that section). This can be useful when analyzing measurements over a longer period and an overall HotFile picture is needed.

The HOTFILES database layout is:

| Field | Type | Width | Decimal |
|---|---|---|---|
| F_NAME | C | 80 | 0 |
| OPEN_CNT | N | 10 | 0 |
| READ_CNT | N | 10 | 0 |
| WRITE_CNT | N | 10 | 0 |
| TOTAL_CNT | N | 10 | 0 |

# "Summary" display description

The first part gives some general information about the dump file such as the filename, the start time and end time and the elapsed time.

```
Capture start at 11-NOV-1993  20:45:06

Summary:

   Input file: TEST.DMP

   Start time: 20:45:06
   End Time:   23:52:58

   Elapsed Time: 3 hour, 7 min, 52 sec   (11272 sec)
```

There is a full time description every time the capture is enabled or re-enabled. The elapsed time is computed by withdrawing the capture interruptions.

```
  FCB SearchFirst        13
  FCB SearchNext        146
  FCB Delete              1
  Create File           109
  Open File             552
  Real Close File       659
  Dummy Close File      445
  Delete File            15
  Lseek                1417
  Execute file           17
  Find First           1714
  Find Next             375
  Seq Large Read       1912     Average size:   4107 Bytes
  Rnd Large Read        590     Average size:   4389 Bytes
  Seq Small Read       4961     Average size:   1900 Bytes
  Rnd Small Read        427     Average size:   1396 Bytes
  Seq Large Write       673     Average size:   4766 Bytes
  Rnd Large Write       198     Average size:   8005 Bytes
  Seq Small Write       780     Average size:   1461 Bytes
  Rnd Small Write        30     Average size:    721 Bytes

  TOTAL             15035
  IOs per second     1.33
```

This section lists the measurements for all the categories of IO monitored where there is a non zero result. The "Total" is the sum of all IOs. The "Number of IOs per second" is a very good indication of the IO load of the station analysed.
A "dummy close" is a close that is automatically performed by DOS at a program exit: it closes all file handles even if no file is assigned.
The sizes shown for each Read/Write category is an average size. The small Read/Write size is from 1 byte to 4000 bytes. Those informations can be useful to determine cache buffer size (block size).

```
List of executed files
    20:45:37.8   C:\BORLANDC\BIN\BC.EXE
    20:45:41.7   C:\BORLANDC\BIN\DPMILOAD.EXE
    20:48:37.9   C:\COMMAND.COM
    22:13:05.0   C:\COMMAND.COM
    22:13:09.6   C:\JMM\IO_SNOOP\IO_SNOOP.EXE
(...)
```

This part lists all the files that have been executed during the monitoring period. This can be interesting when trying to know what applications are used by typical users.

```
Small Read and Write repartition            (Size smaller then 4000 bytes)

           Seq Read   Rnd Read  Seq Write   Rnd Write
<   100       1054         71        232          20
<   200         41         28          0           0
<   300         72          6          0           0
<   400         40         22          0           0
<   500         38         13          0           0
<   600         76          9          1           0
(...)

Large Read and Write repartition            (Size larger then 4000 bytes)

           Seq Read   Rnd Read  Seq Write   Rnd Write
<  5500       1911        575        657         180
<  7000          0          6          0           0
<  8500          0          3          0           0
< 10000          0          0          0           0
< 11500          0          0          0           0
< 13000          0          0          0           0
(...)
```

This last pair of list gives a "Gauss repartition" of the average size of an IO. In each column is the number of times an IO was made with a size within the boundaries of the line (more then the previous line and less then the value of the line). Each line represents 100 bytes for the "small" part and 1500 bytes for the "large" section.
Analyzing this data can be useful to adjust the average size computed to something more realistic.

# Analyse Strategy

Beside its use as a debugging tool, IO_SNOOP can be useful for "knowing" the load of a network. A system administrator should have an accurate idea of the type of workstation and workstation usage that make up the network he's responsible of.

The first step is to classify the users of the network. From a list of connected workstations and users, create a typology of usage such as "Secretary", "Terminal Emulation", "Programmer", etc. Knowing the proportion of user in each category, you can build a statistically correct sample and look for volunteers. IO_SNOOP will lightly slow down the operation and the dump files can get big! So be sure to have enough disk space so that the user will not be penalized for participitating in the test.

For the statistical quality of the study, the samples should be taken on a period long enough, for example two weeks. It is best to have one separate dump file for every day: IO_LIST could have unexpected result for dump files spanning several days.

Here after is a sample batch file that can be added in the user startup procedure. It uses a small utility, STRDATE, that sets an environment variable with the date. This variable can be use to create a different file everyday.

```
@echo off
rem Set date as an environment variable
c:\io_dump\strdate
rem If there is already a dump file for today don't compress it
if exist c:\io_dump\%_strdate%.dmp goto done
c:
rem Compress and remove the previous dump
c:\io_dump\pkzip -m c:\io_dump\%NODENAME%_IO.ZIP c:\io_dump\*.DMP
:done
rem Restart the capture in a file having the date as name
c:\io_dump\IO_SNOOP c:\io_dump\%_STRDATE%.DMP
```

The biggest work is now the analyse of that data. Consolidation should be made on a spreadsheet and by carefully examining the " gaussian" repartition of the data. The end result is the setup of a characteristic file that could be used as an input file to IO_SIMUL for the second part of the study of the network's behavior.

# 4

# IO_SIMUL

This program will allow the user to accurately simulate and time the IO usage of a PC. A set of benchmark is runed with values representing the PC load. Those values are gathered with the help of IO_SNOOP / IO_LIST. Measures can be run on a set of drives, at a certain interval and specified number of times.

This simulation can be useful to measure response time variations with the increase of the number of normalized PC on particular server setup. Drawing a graph with those figures can help in "Capacity Planning" efforts. It will reveal informations like the pivot number of PC where performance starts to degrade rapidelly.

An other useful information is having an accurate and realistic tool to measure network response time and being able to assess correctly "impressions" of unusual network performance.

Usage:

```
IO_SIMUL Drives [Drives] [@response file] [switches]
```

| | | |
|---|---|---|
| drive: | | directory specification to benchmark |
| @file | | specifies a response file name (list of commands) |
| /? | | Prints this help text |
| /interval=min | | Interval in minutes between benchmarks |
| /count=n | | Number of times the benchmark must run (0 = forever) |
| /logfile[=file] | | specifies an output file (default: IO_SIMUL.LOG) |
| /Spreadlog[=file] | | specifies a spreadsheet import (default: IO_SIMUL.PRN) |
| | | |
| FileSize = n | | Size of the test file |
| BigReadCnt | = n | Number of big buffer reads |
| BigReadSize | = n | Size of " |
| BigWriteCnt | = n | Number of big buffer writes |
| BigWriteSize | = n | Size of " |
| SmallReadCnt | = n | Number of small buffer reads |
| SmallReadSize | = n | Size of " |
| SmallWriteCnt | = n | Number of small buffer writes |
| SmallWriteSize | = n | Size of " |
| RandBigReadCnt | = n | Number of Random big buffer reads |
| RandBigReadSize | = n | Size of " |
| RandBigWriteCnt | = n | Number of random big buffer writes |

| | | |
|---|---|---|
| RandBigWriteSize | = n | Size of " |
| RandSmallReadCnt | = n | Number of Random Small buffer reads |
| RandSmallReadSize | = n | Size of " |
| RandSmallWriteCnt | = n | Number of random Small buffer writes |
| RandSmallWriteSize | = n | Size of " |
| FindFirst | = n | Number of first file searches |
| FindNext | = n | Number of subsequent file searches |
| Create | = n | Number of file creations |
| OpenClose | = n | Number of file Open and Close operations |

Note: commands are not case sensitive and can be abbreviated up to the point where it becomes ambiguous.

# Parameters description

**Drive:** A drive letter or a series of drive letter to be tested (delimited with a space). The drive letter can be a file path. During the benchmark, a number of temporary files will be created.

**@file** will execute the list of valid IO_SIMUL command contained in the file. It can contain a single or multiple commands per line. A line starting with ";" is considered as a command. Such a list can be generated with IO_LIST and the command /SIMULATOR.

**/?** Displays a list of valid commands.

**/interval=min** Specifies the interval, in minutes, between benchmarks set. A set is all requested test on all drives. The default value is 0 minutes.

**/count=n** Number of times the benchmark set must run. The default value is is 1. If a value of 0 is specified, the benchmark will run forever, or until it is interrupted by <CTRL><C>. The user is then asked if he really wants to quit the benchmark.

**/logfile[=file]** Specifies the name of an output file. The default file name is IO_SIMUL.LOG. In the case that the file already exists, the user has the choice between creating a new one, appending to the existing or quitting the program.See next section for a commented example.

**/Spreadlog[=file]** Specifies the name of a delimited ASCII file that can be imported in a spreadsheet. The default filename is IO_SIMUL.PRN. In the case that the file already exists, the user has the choice between creating a new one, appending to the existing or quitting the program.See next section for a commented example.

**[values]** List of values to be used during the benchmarks. If none is specified, the default values are taken. They represent a typical Windows user load with a test file of 1.5 Mb. This test file size is larger then the default PC cache size (1 Mb). It is generally recommended to turn off caching to avoid uncontrollable influence although you might just want to measure your cache system.

The default values:

DEFAULT VALUE

# Output description

While the screen output shows the progress of the benchmark, the "logfile" output is a record of all datas gathered during the test. The first part of the log file is the list of all the parameters used:

```
Benchmark for PC: UNKNOWN

C:  Local drive

Benchmarking protocol: (times in micro seconds)

Total temporary file size:  1500000 bytes
Sequential Big Write   (96 * 4766 bytes)
Sequential Big Read    (273 * 4107 bytes)
Sequential Small Write (111 * 1461 bytes)
Sequential Small Read  (708 * 1900 bytes)
Small Random Write     (4 * 721 bytes)
Small Random Read      (61 * 1396 bytes)
Big Random Write       (28 * 8005 bytes)
Big Random Read        (84 * 4389 bytes)
FindFirst/FindNext     (244 / 53)
Create and Open/Close  (15 / 78)
```

The name of the PC is taken from the eventual NetBios station name (int 21h, Function 5Eh, Subfunction 0). It indicates "UNKNOWN" when no name was found. The benchmarked drives are listed with, if possible, the full redirection informations.

All the values used for this benchmark are listed under the heading "Benchmarking protocol". The record oriented operations are described by the number of operations and the size of the record. The FindFirst-FindNext gives respectively the number of first trial in a file search and the other, the subsequent searches. "Create and Open/Close" is composed of the number of file creation and the number of file Open and Close operations.

There is one block of results for every drive in every benchmark pass.

```
Benchmarking drive "C:\JMM"                          05/12/93 14:43:04

Test file creation                      Elapsed time   4892.82 ms  (366 KB/sec)

Big Buffers Write benchmark             Elapsed time   3908.11 ms  (149 KB/sec)
Big Buffers Read benchmark              Elapsed time   5209.55 ms  (219 KB/sec)
Small Buffers Write benchmark           Elapsed time    935.12 ms  (158 KB/sec)
Small Buffers Read benchmark            Elapsed time   6434.37 ms  (219 KB/sec)
Small Random Write benchmark            Elapsed time    241.59 ms
Small Random Read benchmark             Elapsed time   2719.84 ms
Big Random Write benchmark              Elapsed time   1738.75 ms
Big Random Read benchmark               Elapsed time   6728.02 ms
Path Search benchmark                   Elapsed time   2161.97 ms
Create and Open/Close                   Elapsed time   2077.89 ms

Total Elapsed Time: 37.05   secs
```

After the full path specification of the location where the benchmark was executed and the date/time information, the various elapsed times are displayed. Time are in milliseconds. Measurements are made with a microsecond resolution. Throughput is calculated only for the sequential operation and the test file creation.

This is a sample of a spreadsheet import file with the same results:

```
"Benchmark for PC: UNKNOWN"

"C:  Local drive"

"Benchmarking protocol: (times in micro seconds)"

"Total temporary file size:  1500000 bytes"
"Sequential Big Write   (96 * 4766 bytes)"
"Sequential Big Read   (273 * 4107 bytes)"
"Sequential Small Write   (111 * 1461 bytes)"
"Sequential Small Read   (708 * 1900 bytes)"
"Small Random Write   (4 * 721 bytes)"
"Small Random Read   (61 * 1396 bytes)"
"Big Random Write   (28 * 8005 bytes)"
"Big Random Read   (84 * 4389 bytes)"
"FindFirst/FindNext (244 / 53)"
"Create and Open/Close (15 / 78)"

"Date" "Time" "Drive" "Big Write" "Big Read" "Small Write" "Small Read" "Random Write" "Random Read"
"FindFirst/Next" "Create Open/Close"
"05/12/93" "14:43:42" "C:\JMM" 3908115 5209553 935116 6434367 241592 2719845 1738753 6728023
"05/12/93" "14:45:31" "C:\JMM" 4883348 6943508 983692 8370299 297540 3732805 2071831 8483663
```

# 5

# Appendix

## Miscellaneous programs

Following programs where added as they may be useful during analyse and benchmarking projects.

### STRDATE

This program will add an environment variable, "_STRDATE" containing the date under the form "YYMMDD". This variable can be easily used in a batch file to create a dump file per day.

```
c:\io_dump\strdate
rem If there is already a dump file for today don't compress it
if exist c:\io_dump\%_strdate%.dmp goto done
```

### MERGEDBF

When trying to analyse the most accessed files (HotFiles option) over a longer period, this utility allows to merge different HOTFILE.DBF databases. The file access counters are updated when that file has already been accessed.

```
MERGEDBF Destination DBF [origin DBF]
```

"Destination DBF" is the name of the new database. "Origin DBF" is the name of the database to be merged. If this database is omitted the contend of the first DBF is displayed, sorted by the number of total access.

### IO_DUMP

This program will take a binary dump file and display its contend one record per line in hex format. This was more a debugging tool but could be useful for somebody wishing to write his own dump lister/analyzer.

### Sample Installation Procedure

To be able to install the program easily on a client station, you can use the following procedure and batch files.

First build a self-extracting archive that will contain all necessary files as the startup batch to be placed in the AUTOEXEC.BAT, AUTOUSER.BAT or login procedure. A sample of such a IO_TEST.BAT is shown in the "Analyse Strategy" section.

You can use the following batch file (PAK.BAT) to create this self-extracting archive.

```
pkzip -a io_test.zip IO_SNOOP.EXE STRDATE.EXE IO_TEST.BAT PKZIP.EXE
zip2exe io_test.zip
```

Put the self-extracting IO_TEST.EXE on a floppy along with the following installation batch (INSTALL.BAT).

```
@echo off
cls
echo This procedure will create a directory on drive C: called C:\IO_DUMP
echo You can start the trace automaticaly by adding the line to your M:\AUTOUSER.BAT
echo      CALL C:\IO_DUMP\IO_TEST
echo .
pause
md c:\io_dump
io_test c:\io_dump\
```

# IO_SNOOP Dump File Format

#### Timestamp record

**LFttttTTTT**

L : record length (1 byte)
F : record identifier (1 byte set to 0xFF)
tttt : ticks number corresponding to TT
TTTT : time in time_t format (4 bytes);

### Output for FCB functions :

**`LttttFdffffffffeeeSS`**

L : record length (1 byte)
tttt : Ticks number (4 bytes).
F : Function code (1 byte).
- 0F : FCB OPEN
- 16 : FCB CREATE
- 10 : FCB CLOSE
- 11 : FCB SEARCH FIRST
- 12 : FCB SEARCH NEXT
- 13 : FCB DELETE
- 14 : FCB READ
- 15 : FCB WRITE
- 21 : FCB RANDOM READ
- 22 : FCB RANDOM WRITE
- 27 : FCB RANDOM BLOCK READ
- 28 : FCB RANDOM BLOCK WRITE

d : Drive letter (1 byte).
ffffffff : File name (8 bytes).
ext : File extension (3 bytes).
SS : size in bytes (2 bytes unsigned integer).
- only relevant for READ/WRITE functions (14,15,21,22,27,28)

### Create File / Open File

**`LttttFHZpp...pp`**

L : record length (1 byte)
tttt : Ticks number (4 bytes).
F : Function code (1 byte).
- 3C : CREATE FILE
- 3D : OPEN FILE
- 6C : EXTENDED OPEN/CREATE FILE

H : File Handle (1 byte).
Z : System file tables (SFT) index (1 byte).
pp..pp : Path name (variable length).

### Read File / Write File

**LttttFHZSS**

L : record length (1 byte)
tttt : Ticks number (4 bytes).
F : Function code (1 byte).
3F : READ FILE
40 : WRITE FILE
H : File Handle (1 byte).
Z : System file tables index (1 byte).
SS : size in bytes (2 bytes unsigned integer).

### Close File / Lseek

**LttttFHZ**

L : record length (1 byte)
tttt : Ticks number (4 bytes).
F : Function code (1 byte).
3E : CLOSE FILE
42 : LSEEK
H : File Handle (1 byte).
Z : System file tables index (1 byte).

### Flock

**LttttFHZS**

L : record length (1 byte)
tttt : Ticks number (4 bytes).
F : Function code (1 byte).
5C : FLOCK
H : File Handle (1 byte).
Z : System file tables index (1 byte).
S : Subfunction 0 = LOCK, 1 = UNLOCK

### Exec File / Delete File / Find First

**`LttttFpp...pp`**

L : record length (1 byte)
tttt : Ticks number (4 bytes).
F : Function code (1 byte).
4B : EXEC FILE
41 : DELETE FILE
4E : FIND FIRST
pp..pp : Path name (variable length).

### Find Next

**`LttttF`**

L : record length (1 byte)
tttt : Ticks number (4 bytes).
F : Function code (1 byte).
4F : FIND NEXT