DiskMark Performance Measurement

Jozo Dujmović

Basic disk file operations

1. Create a binary file consisting of records that have a fixed structure (e.g. account number, customer name, and balance)
2. Sequential read of the binary file
3. Random read of the binary file

DiskMark

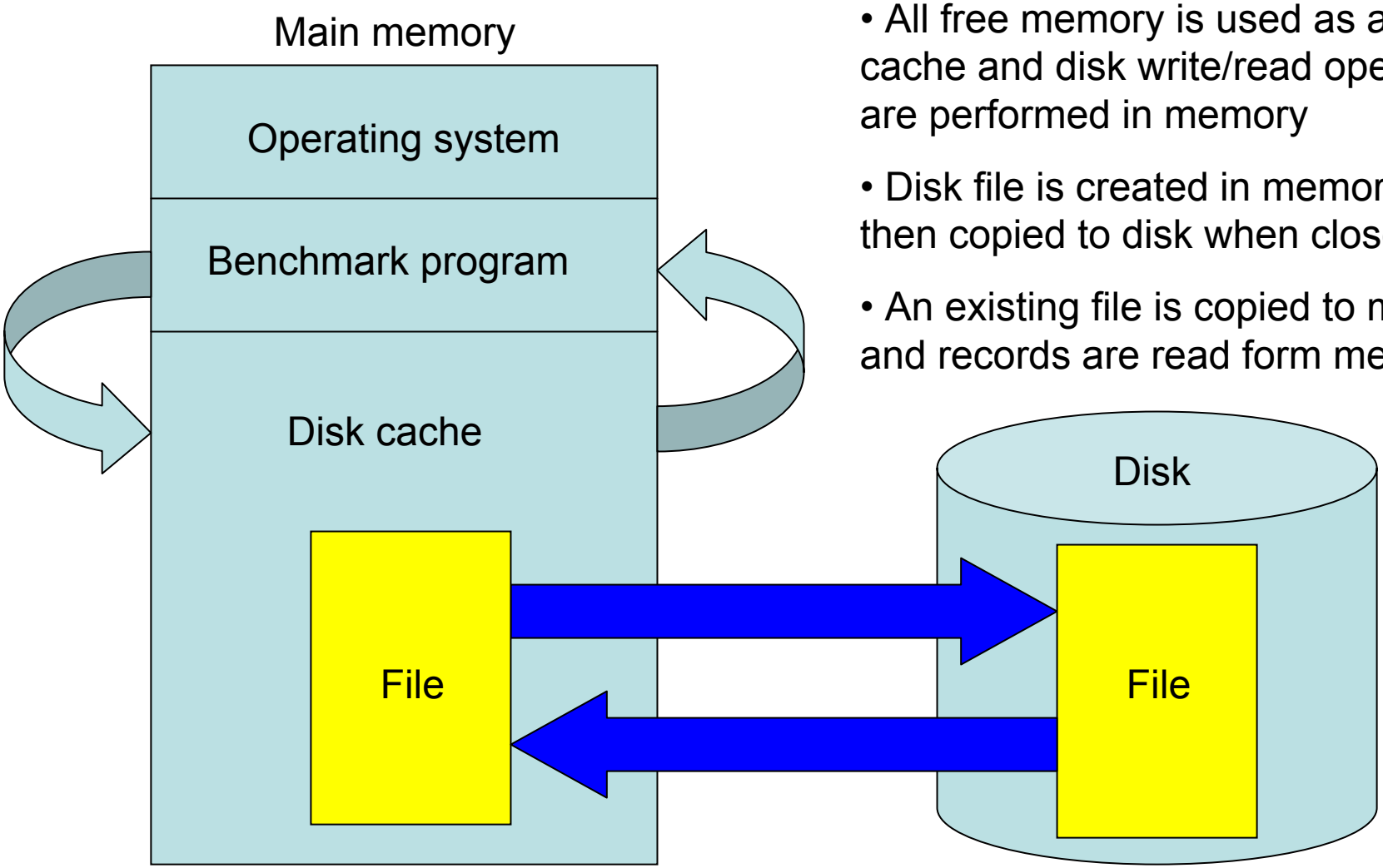
- Measured speed indicators:
  - Vseqw = disk sequential write speed
  - Vseqr = disk sequential read speed
  - Vranr = disk random read speed
- DiskMark is defined as the mean speed of basic disk operations:

  $$Vdisk = 3/(1/Vseqw + 1/Vseqr + 1/Vranr)$$

Parameters of the benchmark program

- Record size
  - Convenient size is a fraction of the 512 byte sector
  - E.g., 512/8=64 bytes
- File size
  - File size can be determined using a fixed file create time
  - File size can be defined as a constant value (number of records)

Using main memory as a disk cache



- All free memory is used as a disk cache and disk write/read operations are performed in memory
- Disk file is created in memory and then copied to disk when closed
- An existing file is copied to memory and records are read form memory

Performance effects

- Using main memory as disk cache dramatically reduces the number of mechanical disk read and write operations
- For files that can fit in memory the importance of seek time and latency becomes negligible
- Performance of disk read/write operations depends primarily on the following:
  - Processor speed
  - Memory and bus speeds
  - Efficiency of cache data access and data buffering algorithms implemented in the file management part of the operating system

Example

- Rotational speed of disk =7200 rpm
- Revolution time = 7200/60 = 120 rev/sec
- Consequently the number of physical random accesses to disk is <= 120 [1/sec]
- Measured number of random accesses per second > 300000 [1/sec]
- Obviously, random accesses are performed in memory, not on disk

The structure of record

```
const int BytesPerRecord=64;

struct client
{
        int account;
        char name[BytesPerRecord-8];
        int balance;
};
```

File open command

- Open for writing:
  ofstream OS(filename, ios::out | ios::binary);

- Open for sequential read:
  ifstream IS(filename, ios::in | ios::binary);

- Open for random read:
  ifstream IS(filename, ios::in | ios::out | ios::binary);

Sequential write

```
client c;
……..
while( condition )
{ // Create client record
  OS.write(reinterpret_cast< const char *>(&c),
                sizeof(c)); // Write to cache
}
OS.close( ); // Write from cache to disk
```

Sequential read

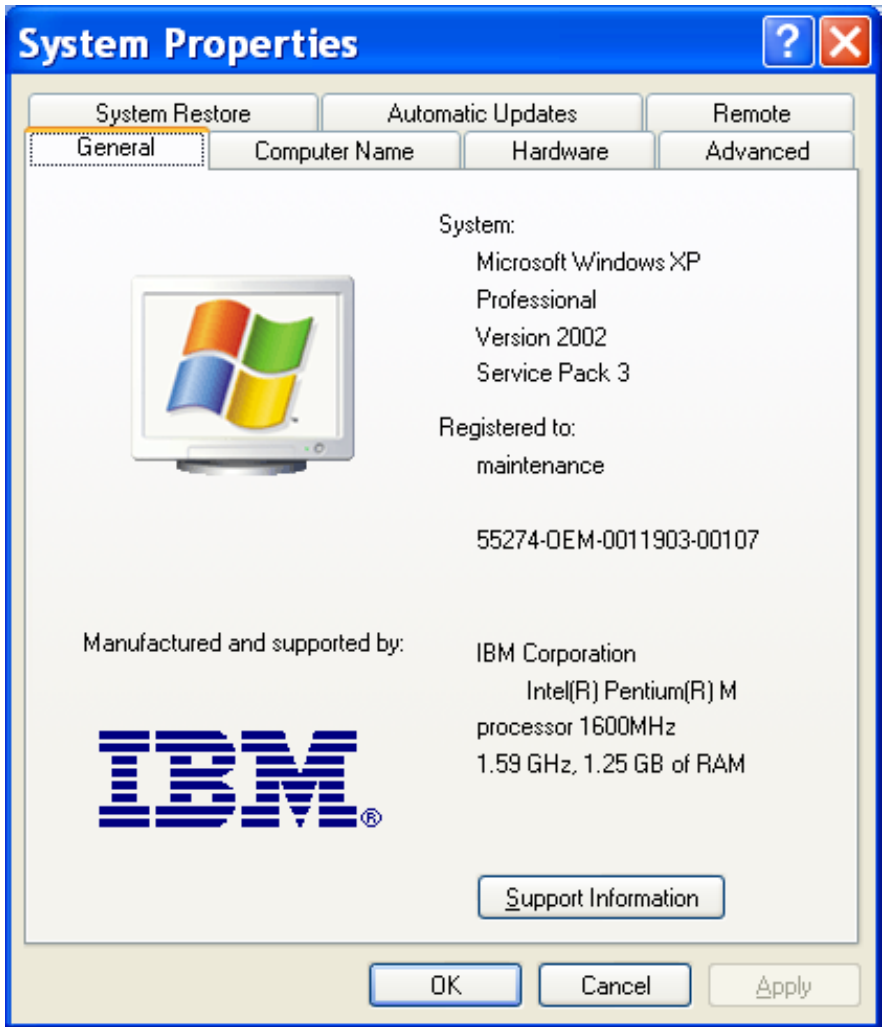
```
while(! IS.eof( ))
{
    IS.read(reinterpret_cast< char *>(&c),
                sizeof(c));  // Read from cache
    // process client record
}
IS.close();
// Display results
```

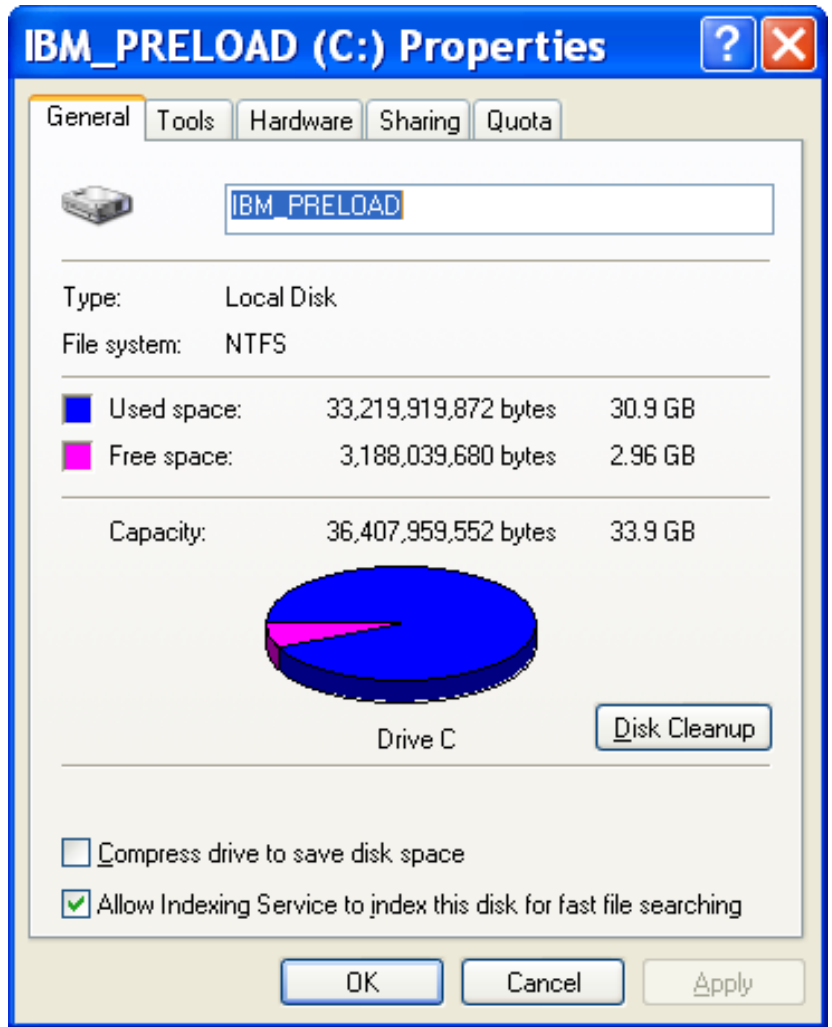
Random read

```
while( condiiton )
{
    record = int((N-2.)*double(rand())/RAND_MAX)
    IS.seekg(record*sizeof(c));
    IS.read(reinterpret_cast< char *>(&c), sizeof(c));
    // Process client record
}
IS.close( );
```

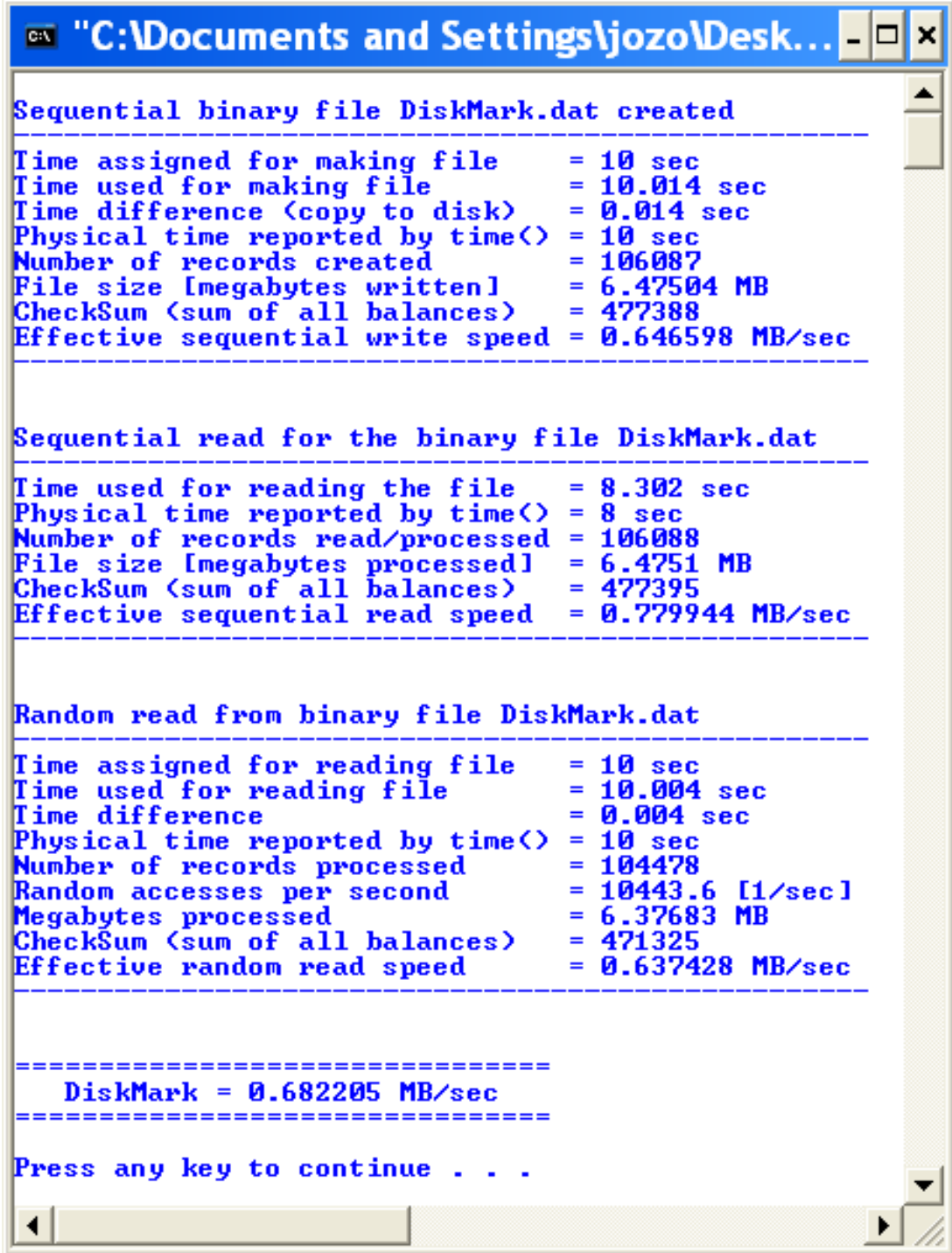
Experiments with a laptop computer

IBM T41, 1.25 GB, Pentium M, Windows XP

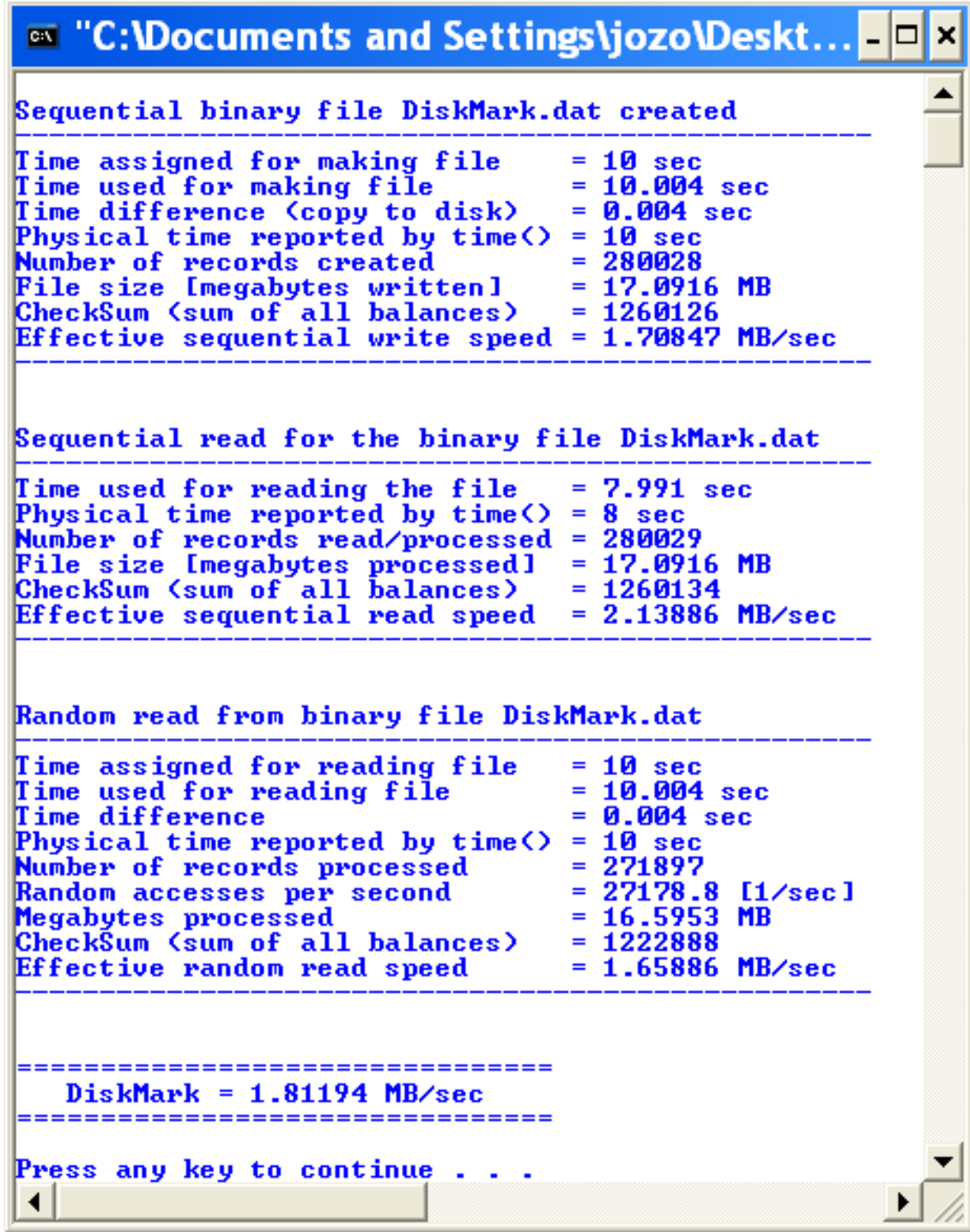




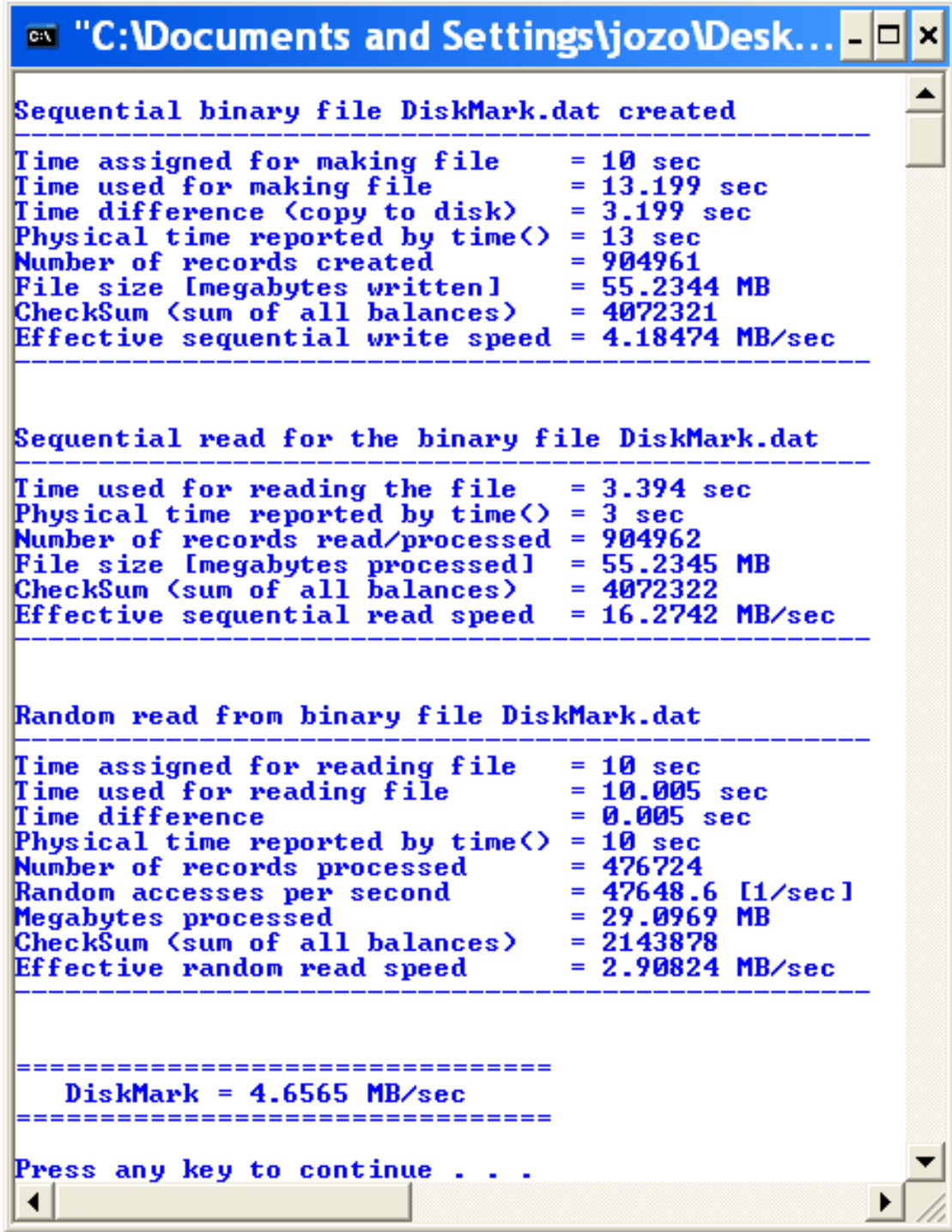
IBM T41
MS VCPP
Debug mode
Battery



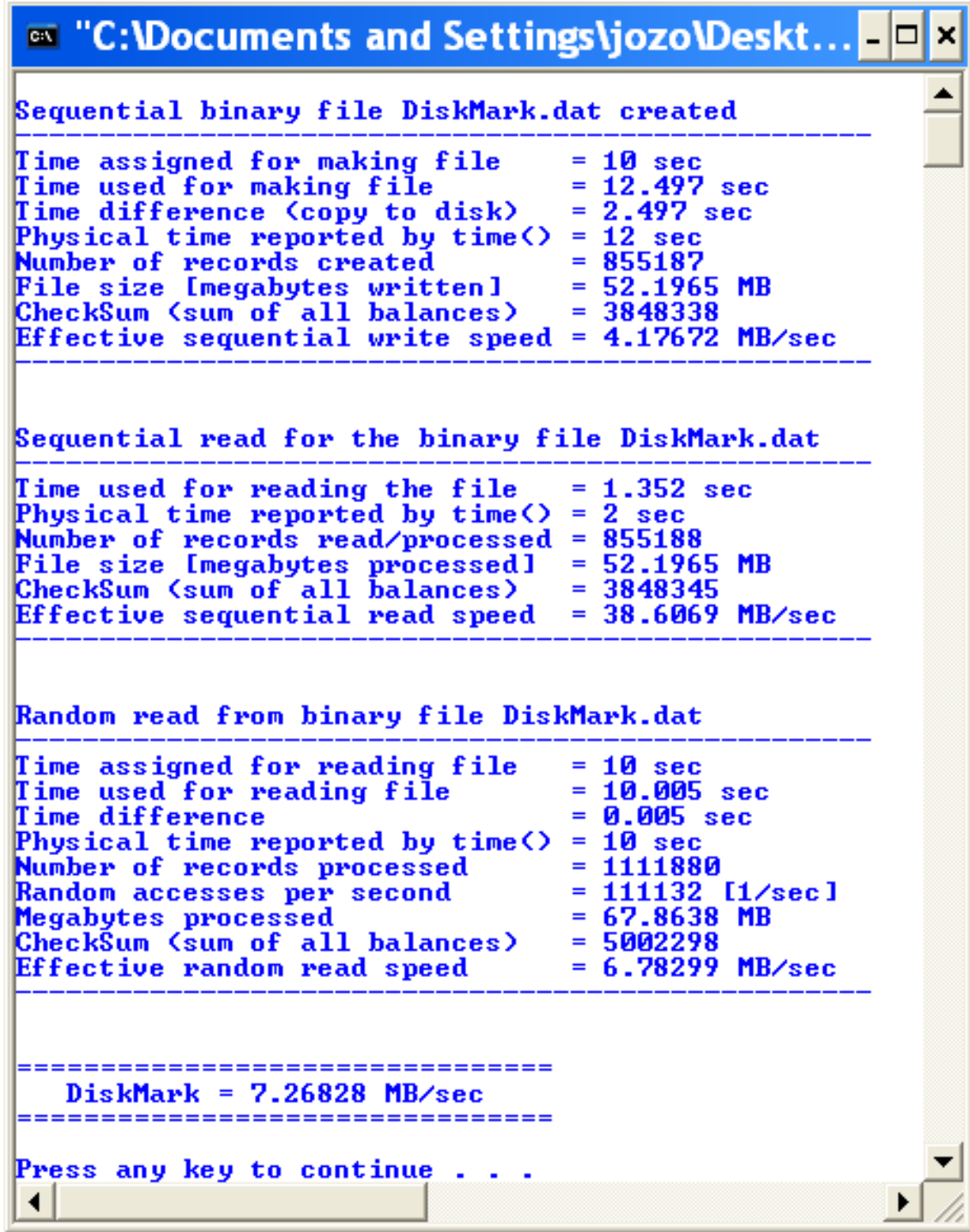
IBM T41
MS VCPP
Debug mode
AC Power



IBM T41
MS VCPP
Release mode
Battery



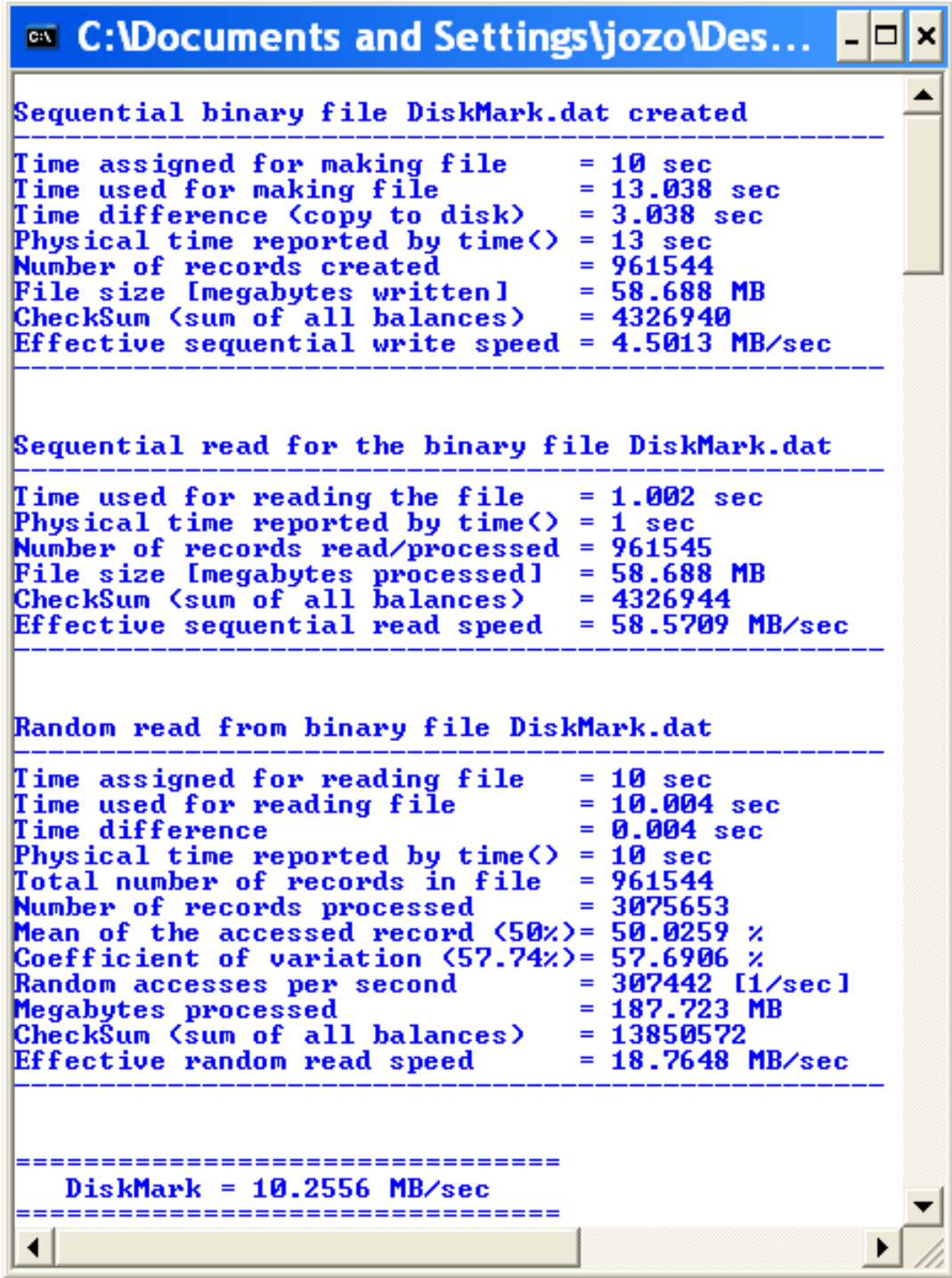
IBM T41
MS VCPP
Release mode
AC Power



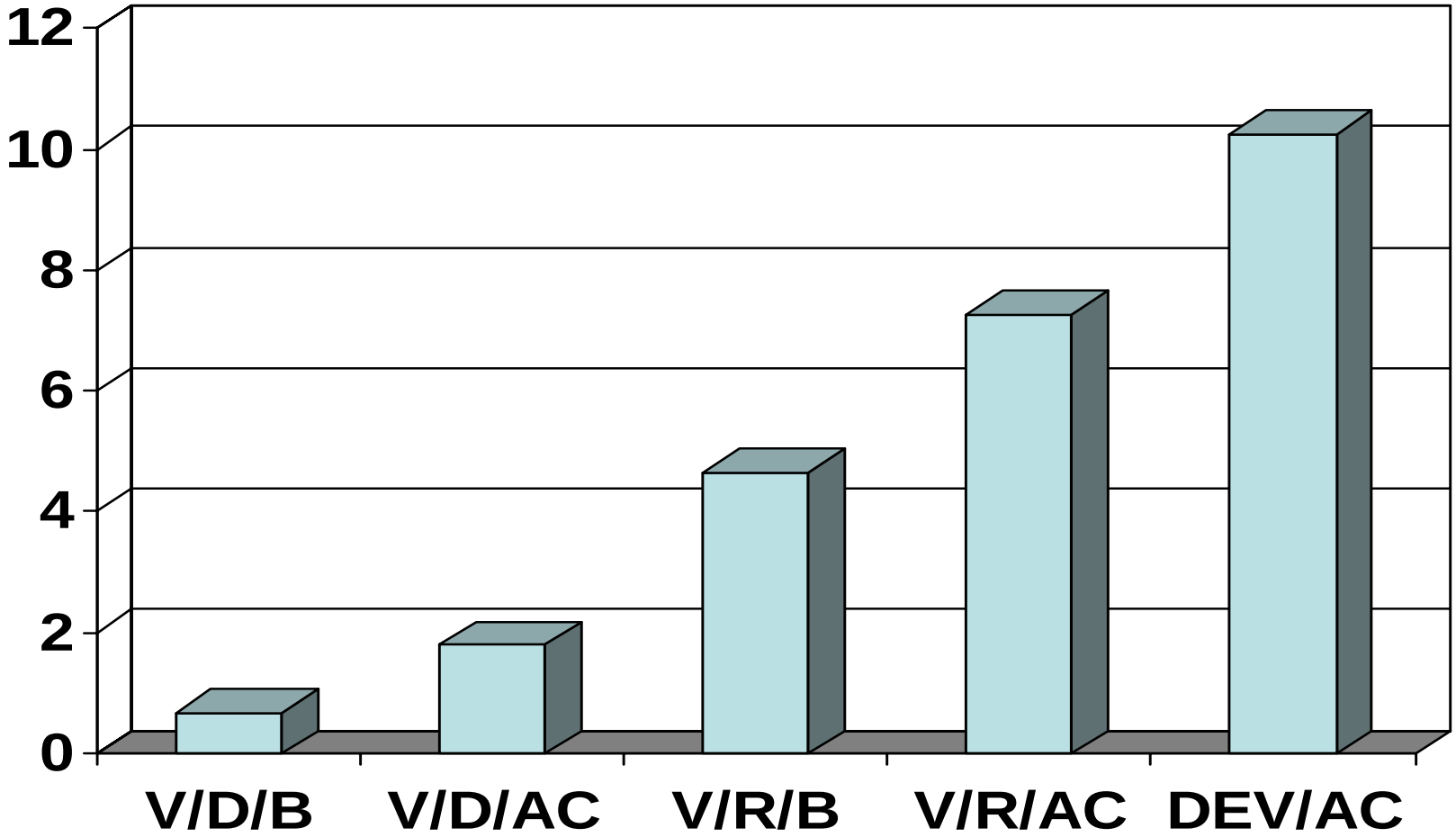
IBM T41
Dev C++
No optimize
AC Power

Note: Experiments with DEV C++ compiler showed no effects of optimization. The default no optimize version generated the best results.

Individual runs generate results that can differ for 10% depending on the available memory.

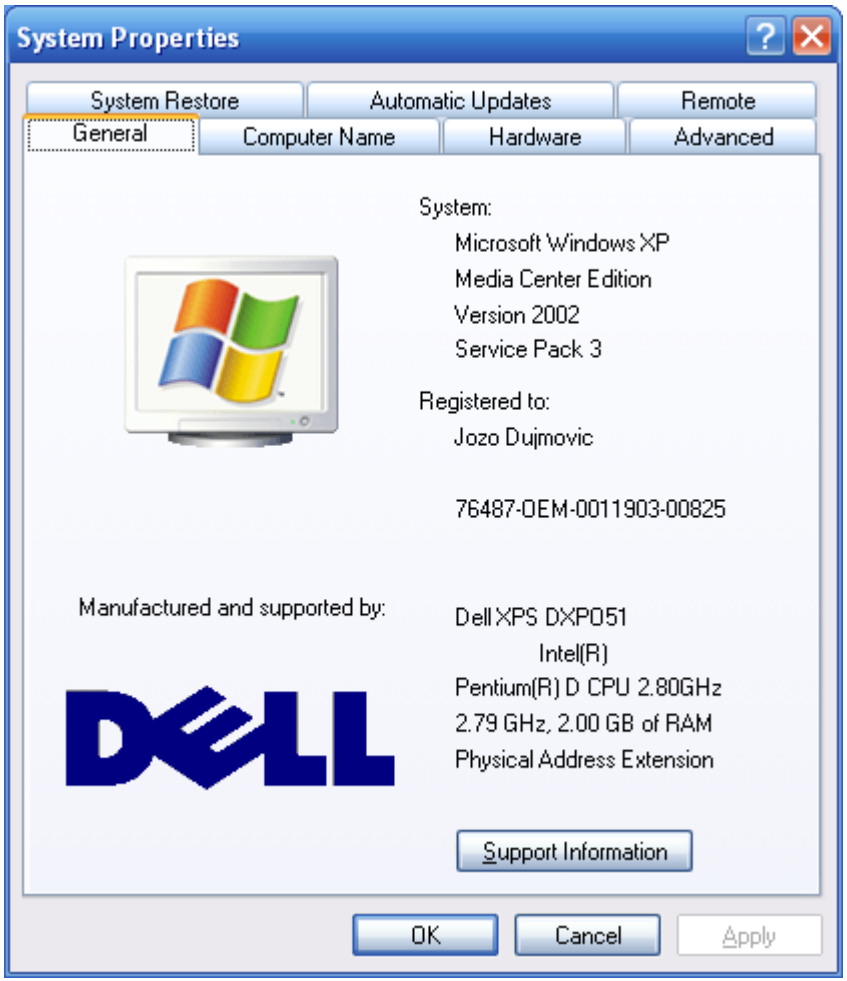


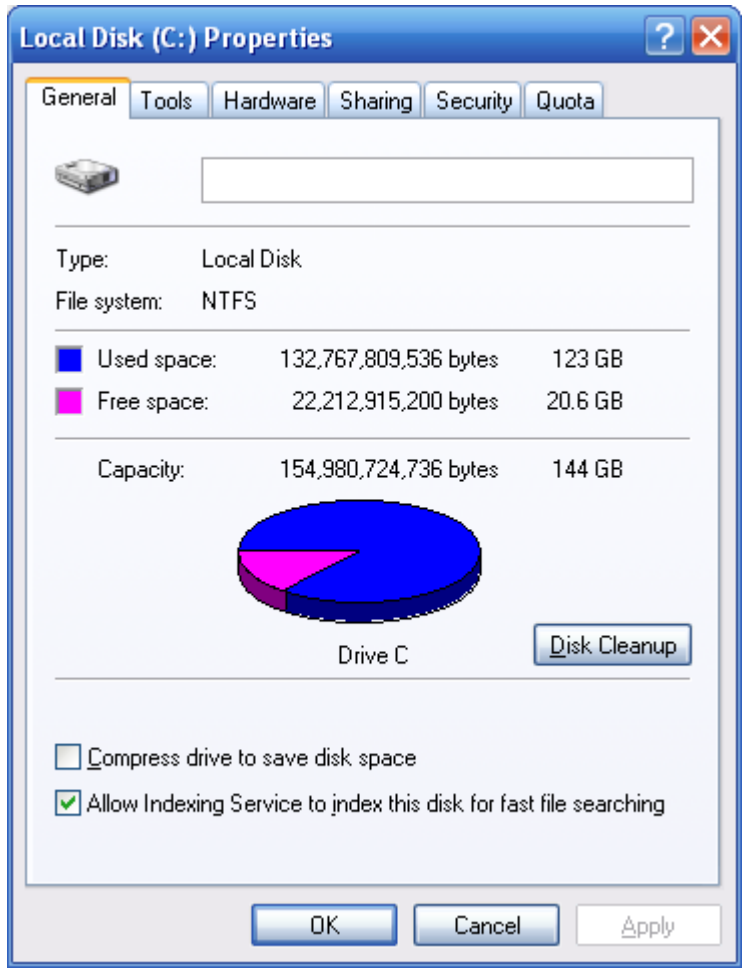
DiskMark Summary for IBM T41



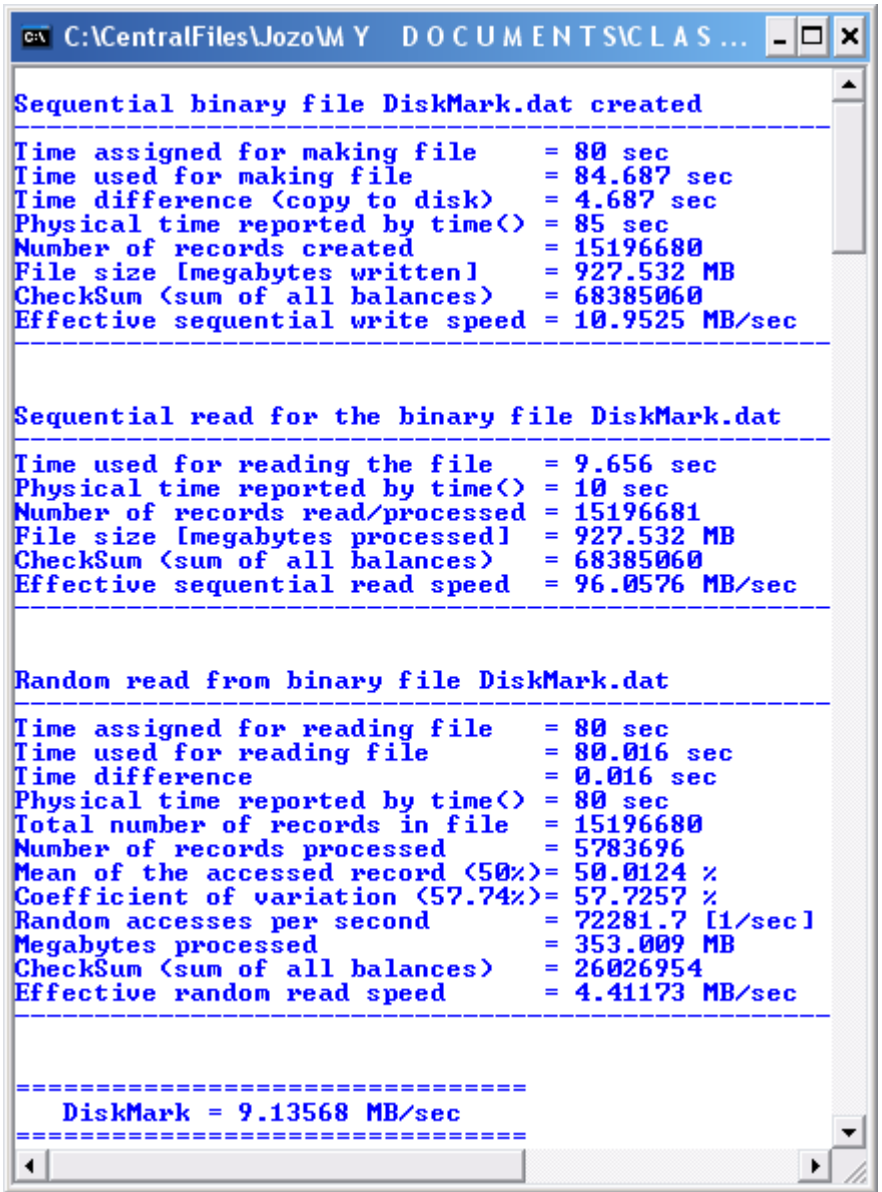
Experiments with a desktop computer

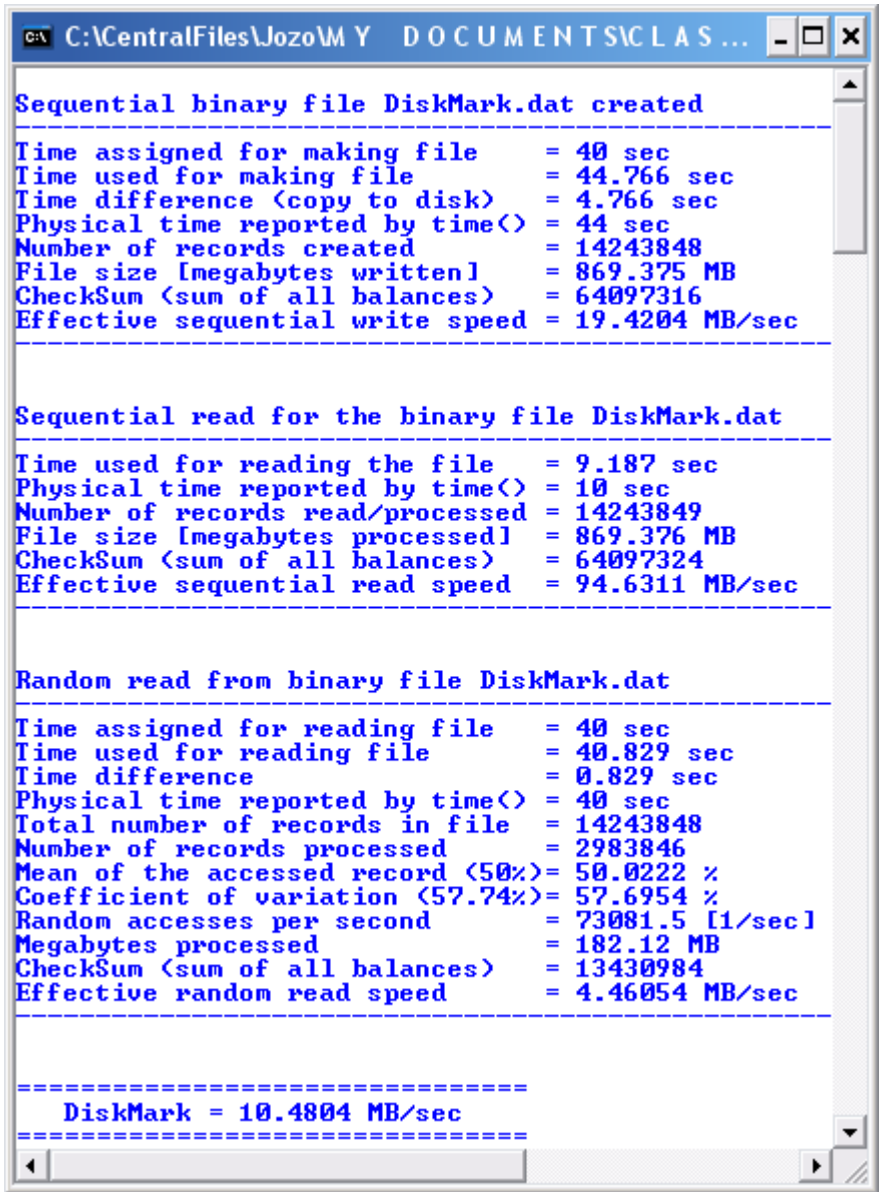
Dell XPS 400, 2GB, 2 cores, Windows XP



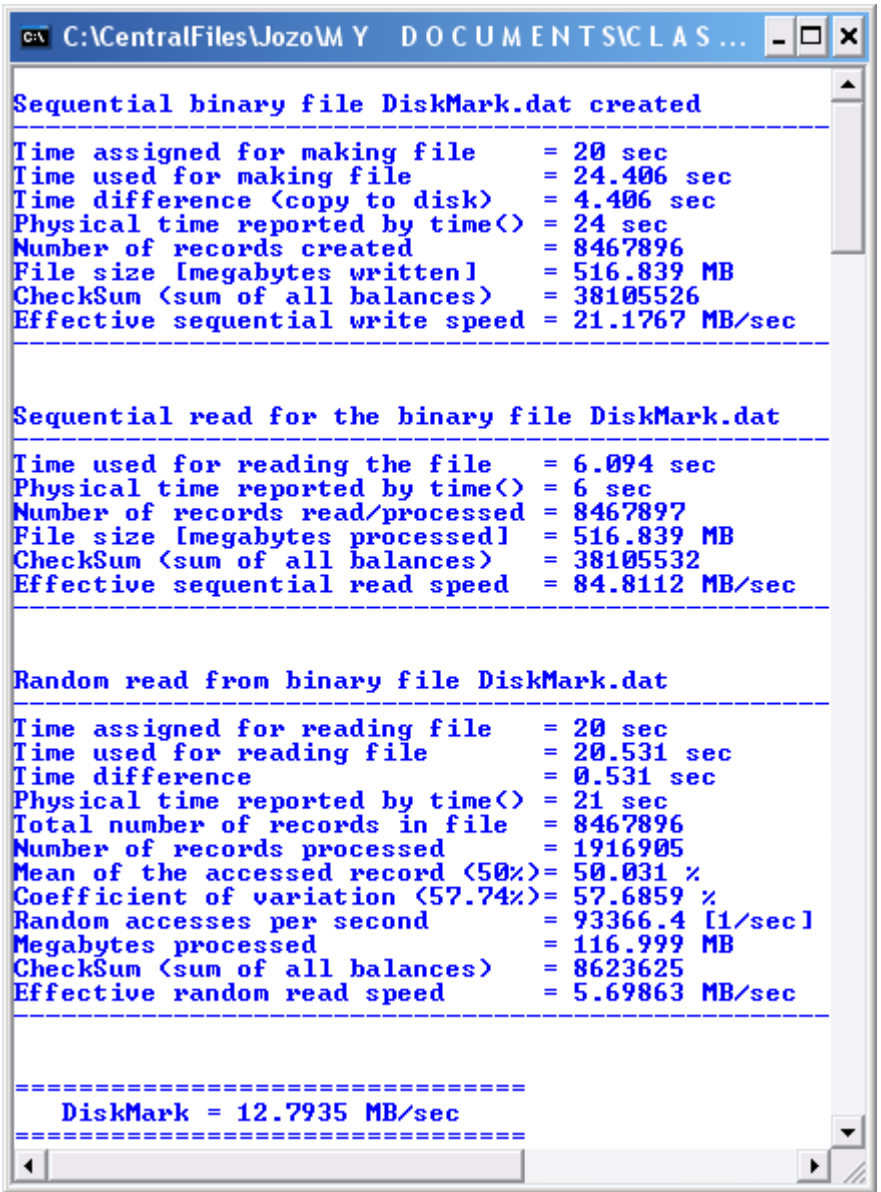


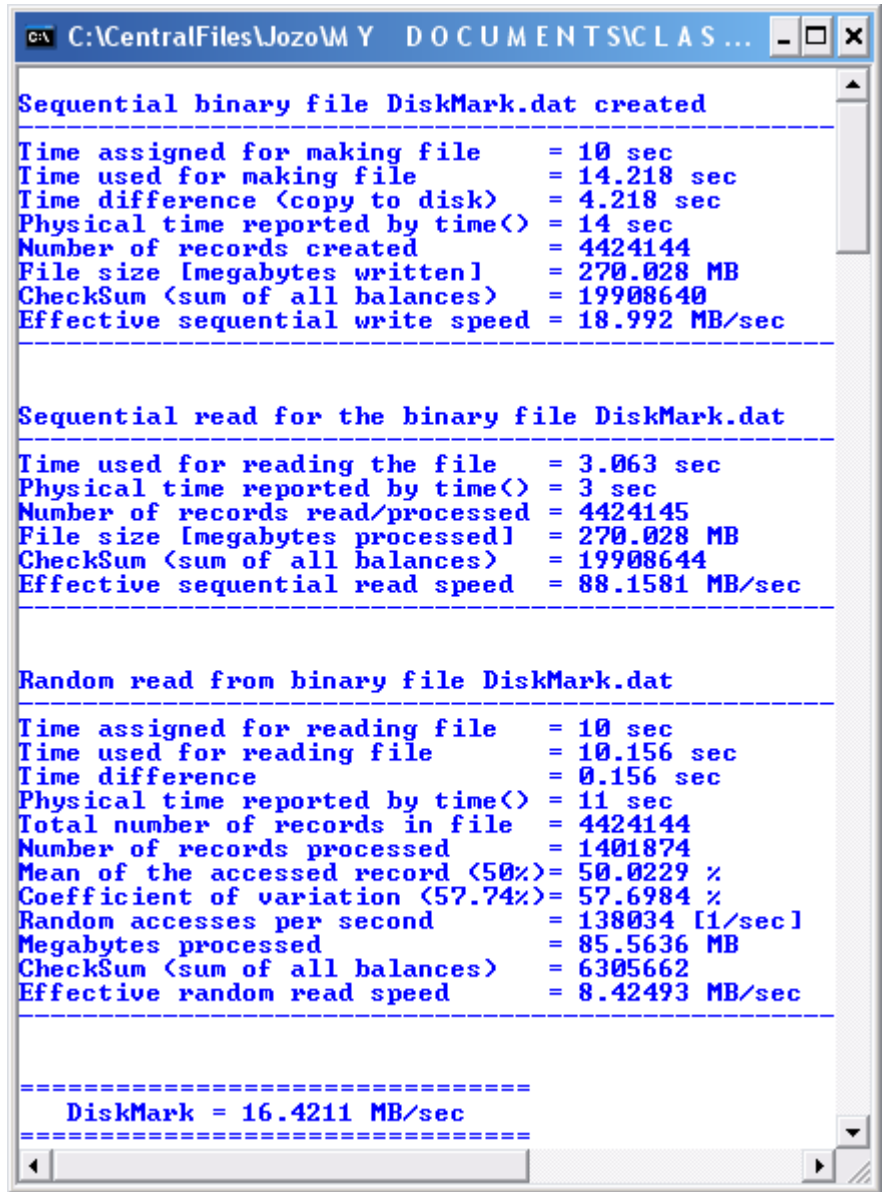
Dell XPS 400 – DEV C++



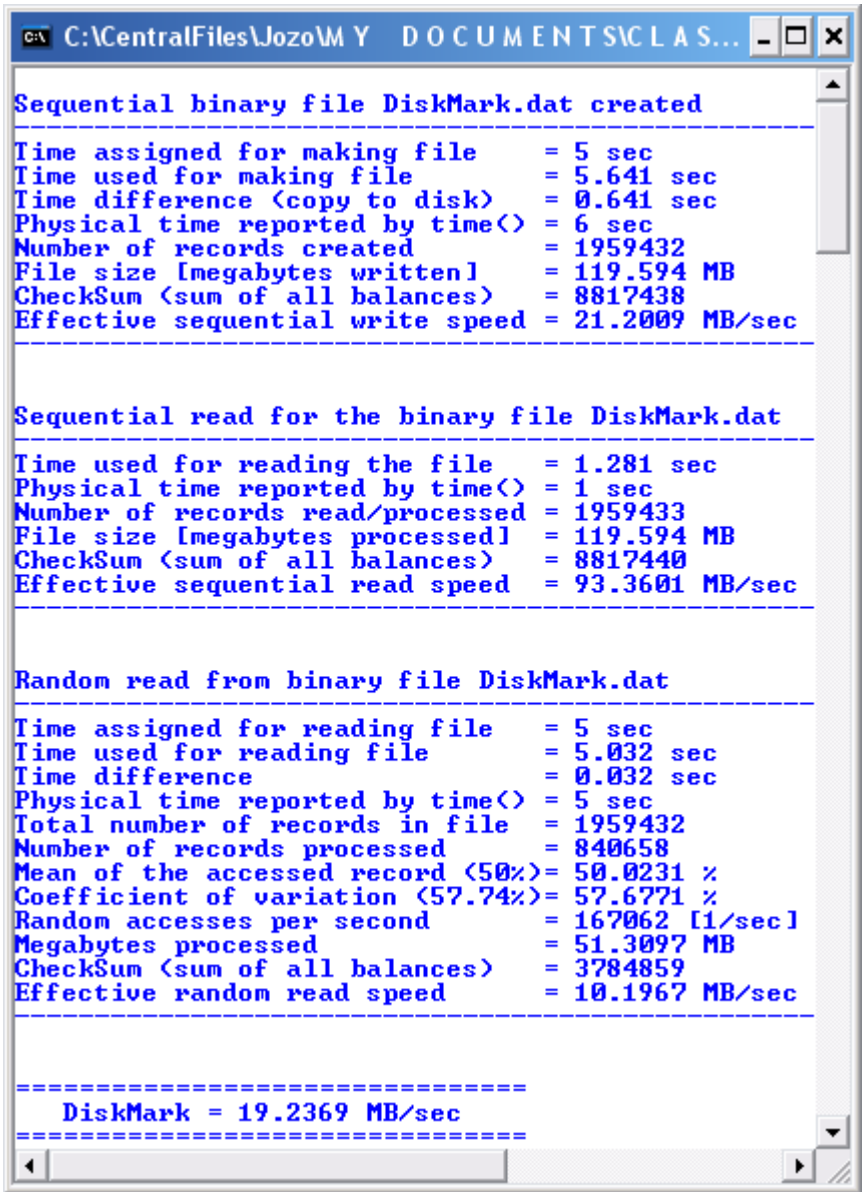


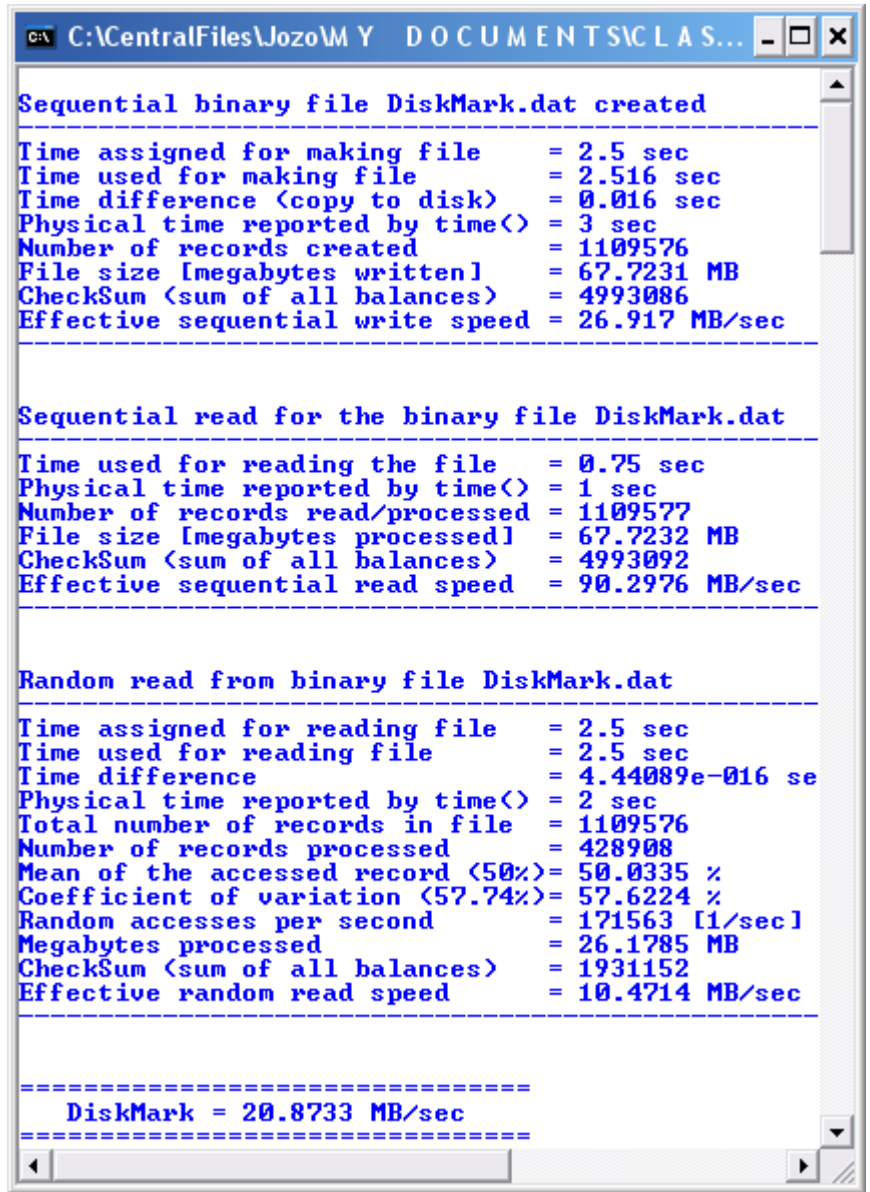
Dell XPS 400 – DEV C++



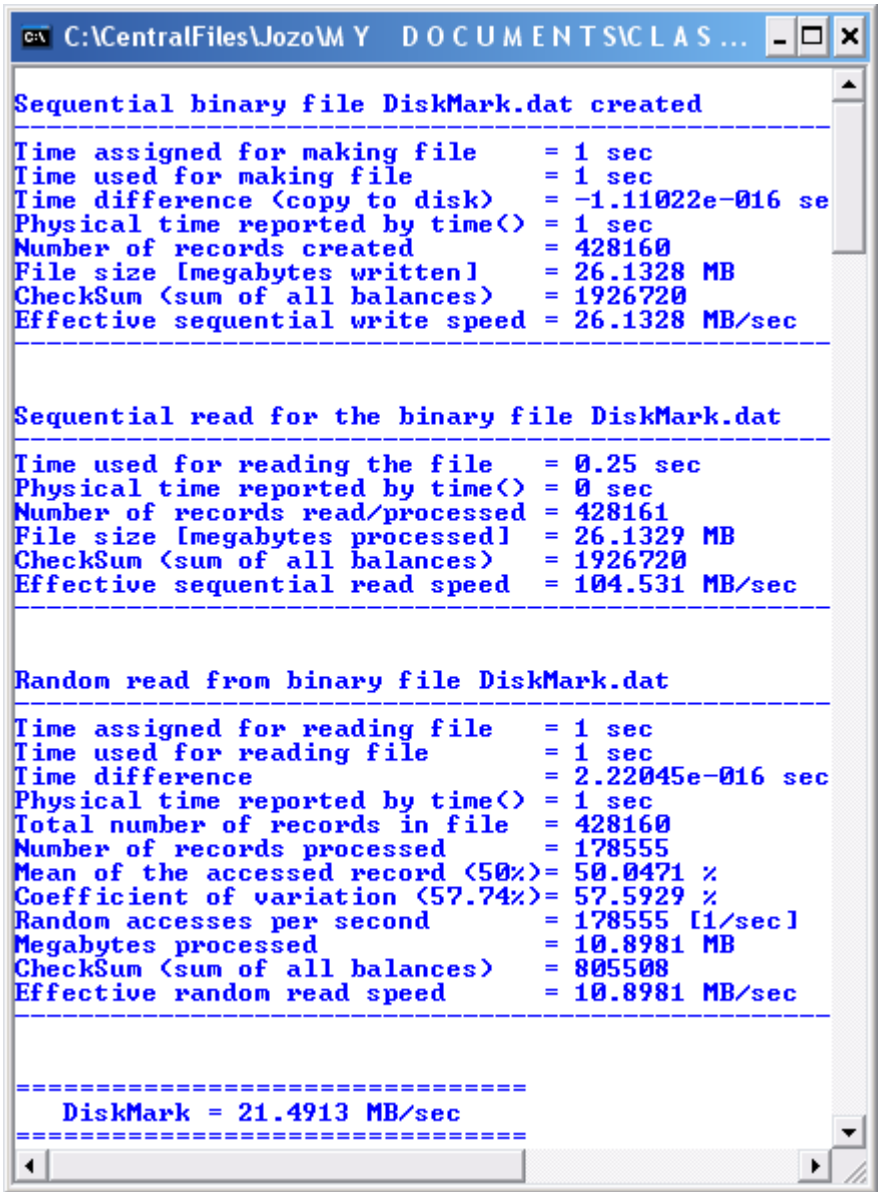


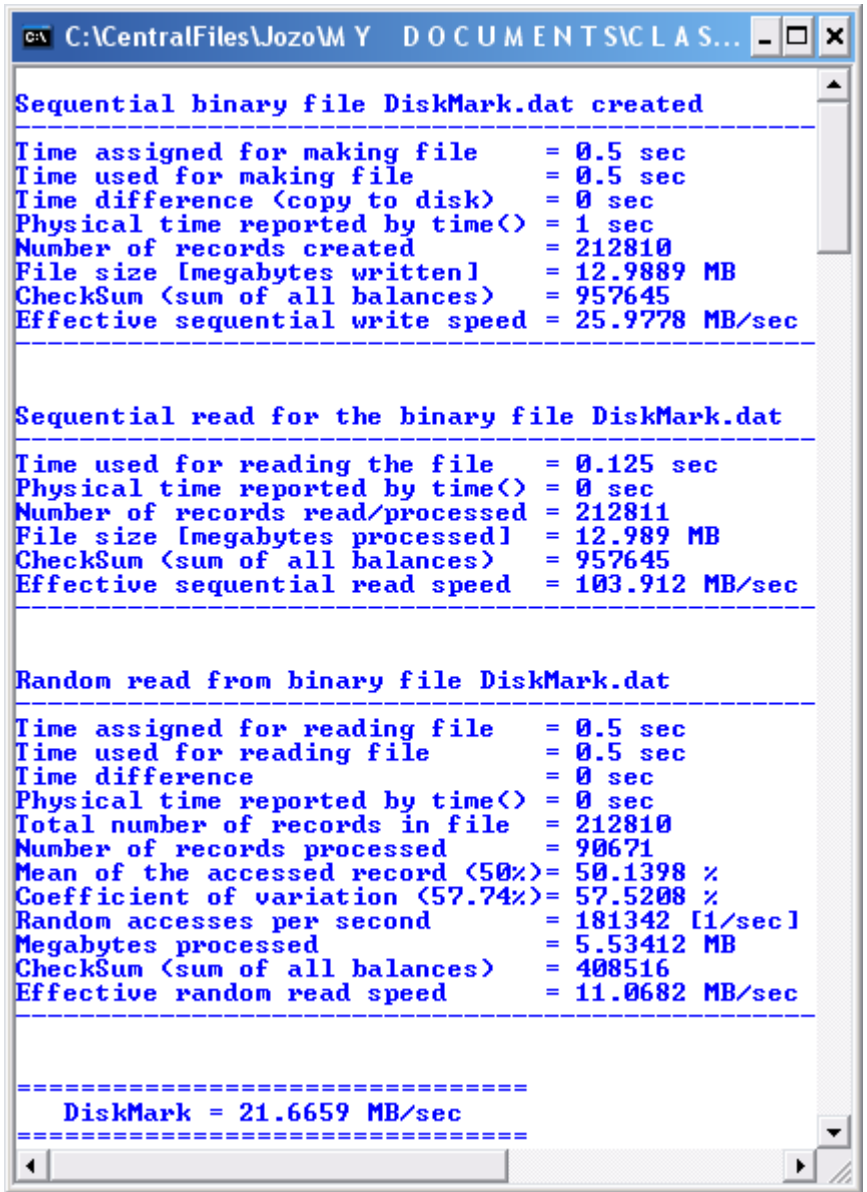
Dell XPS 400 – DEV C++



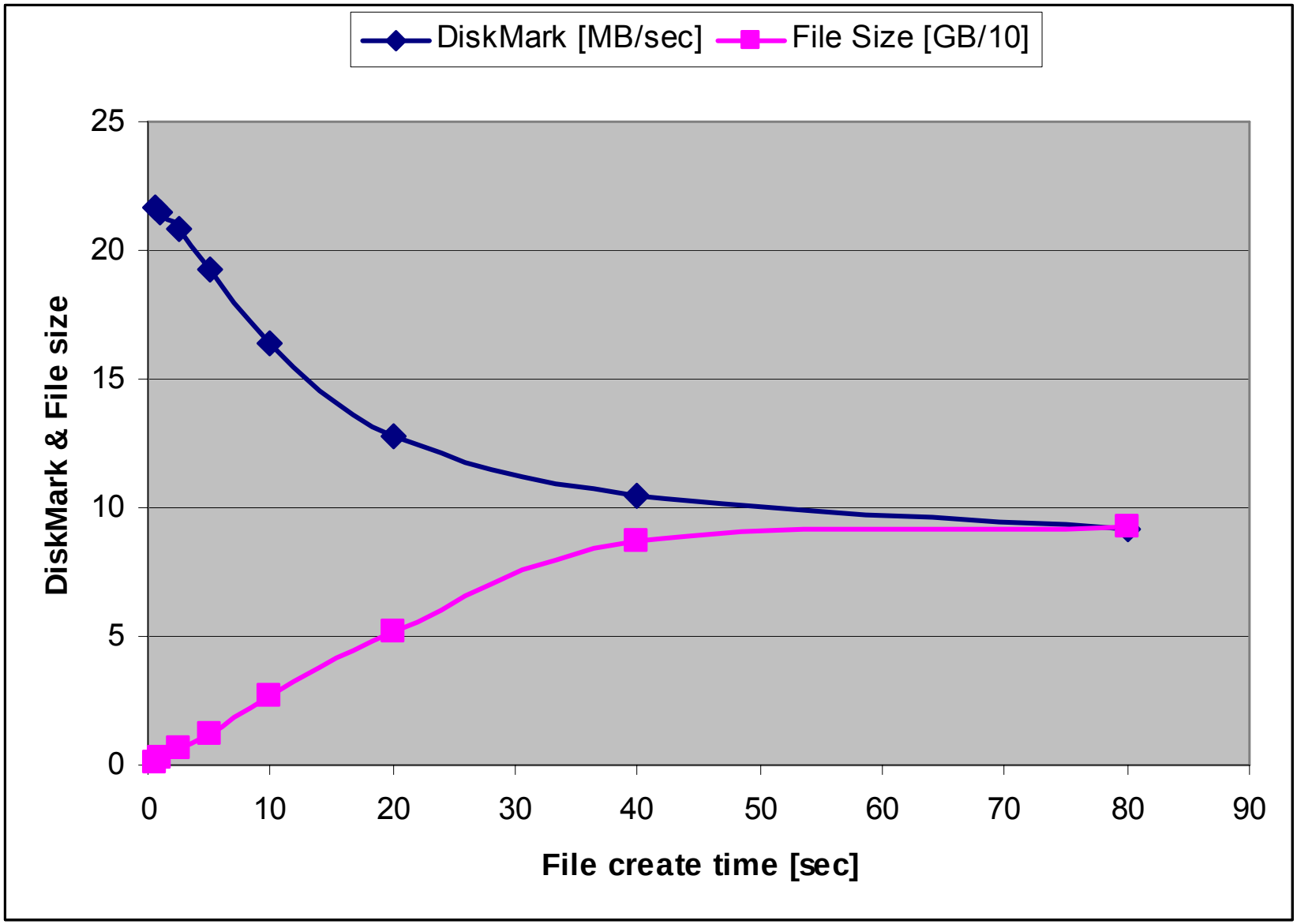


Dell XPS 400 – DEV C++

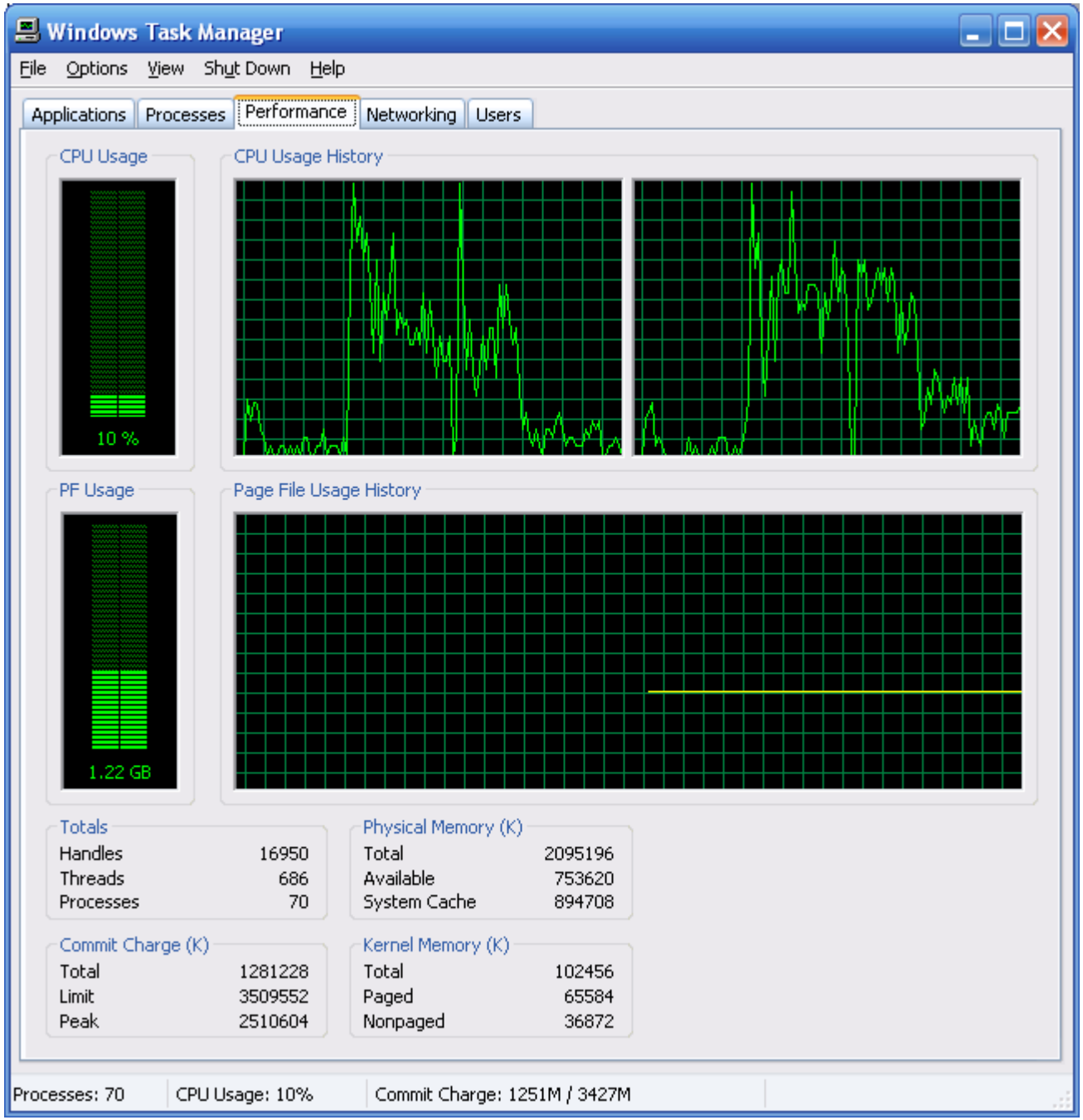




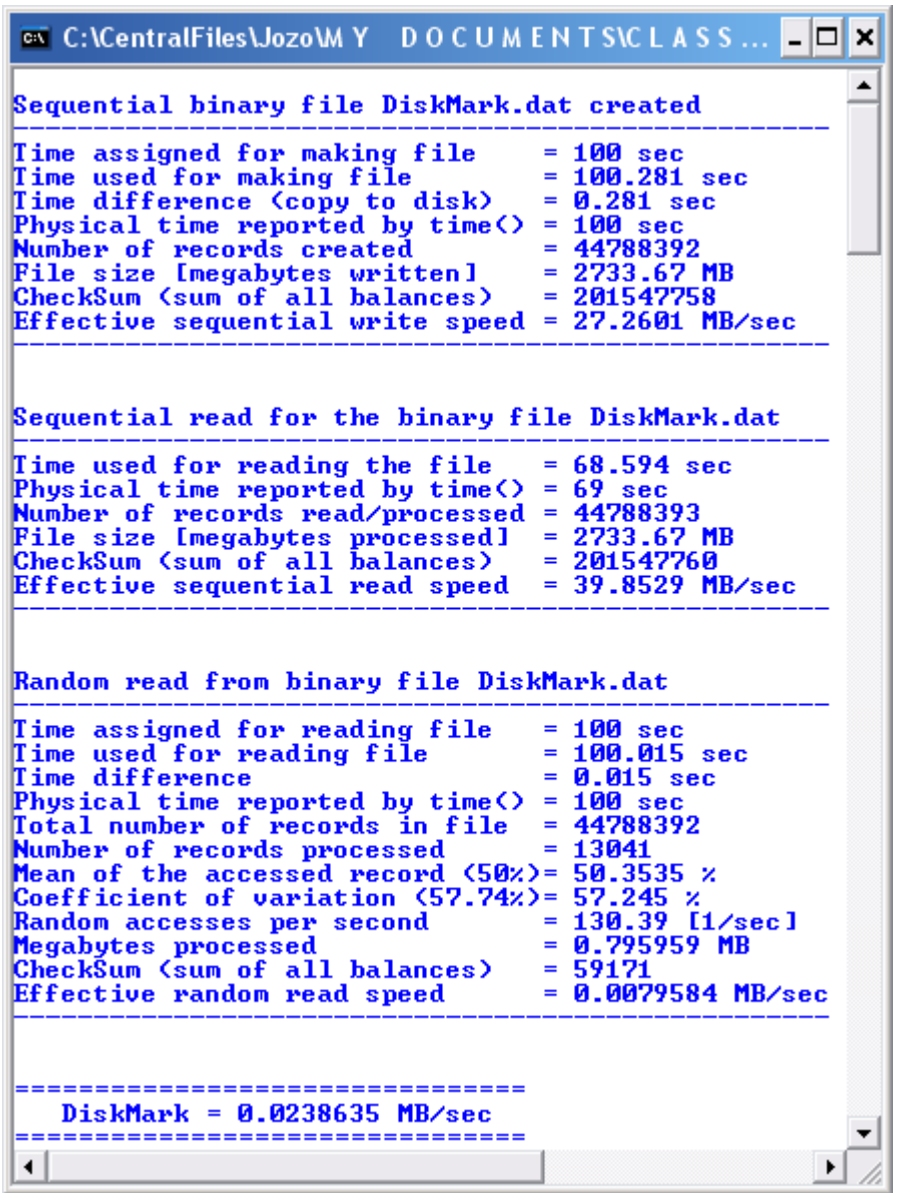
Disk caching effects

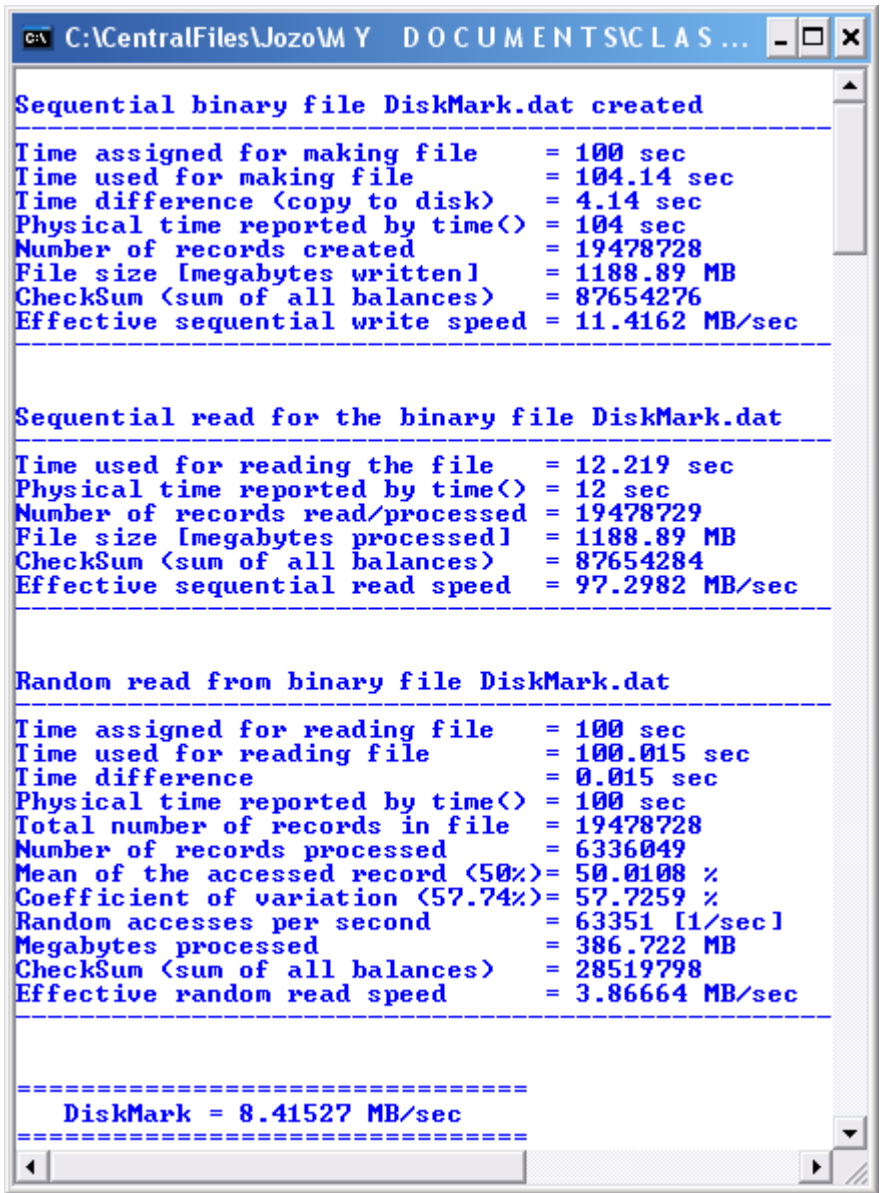


DiskMark Benchmark causes high utilization of processor cores

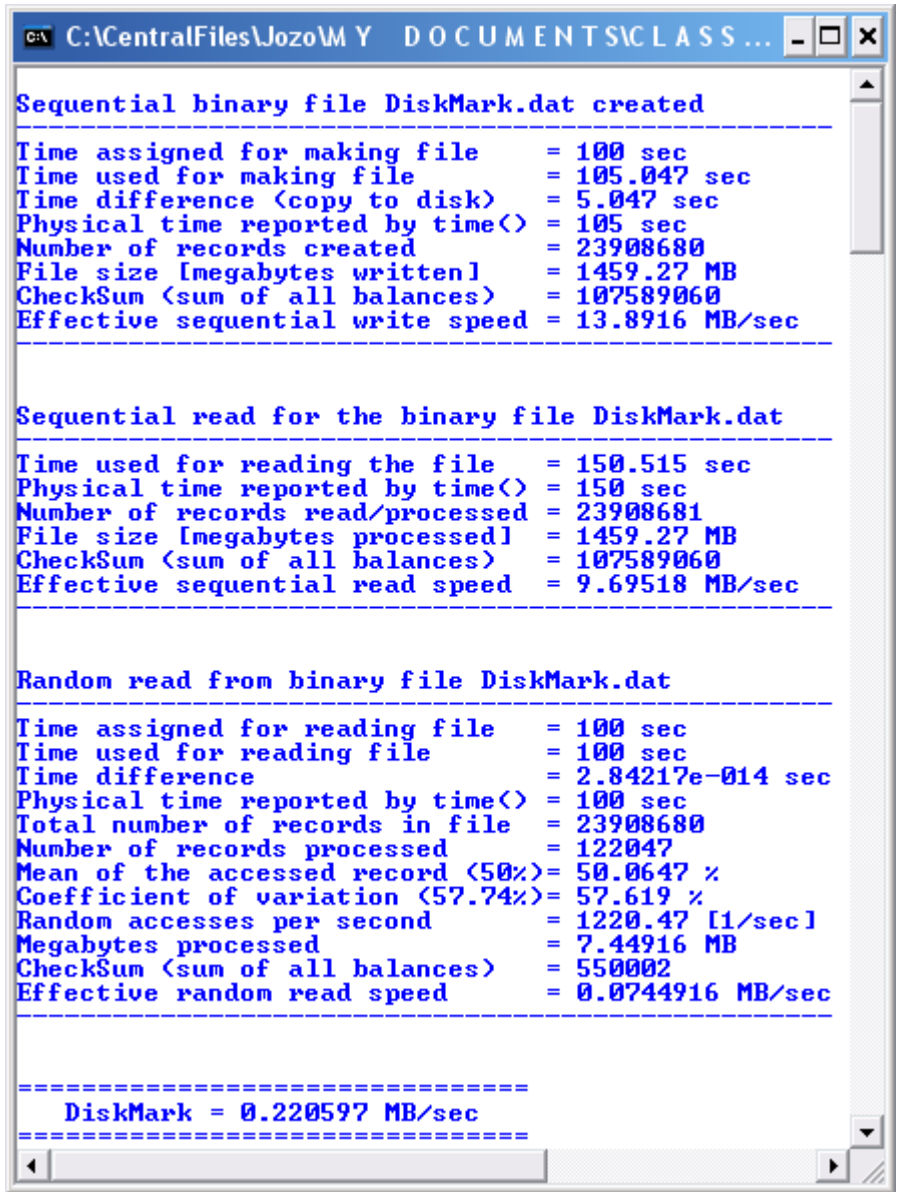


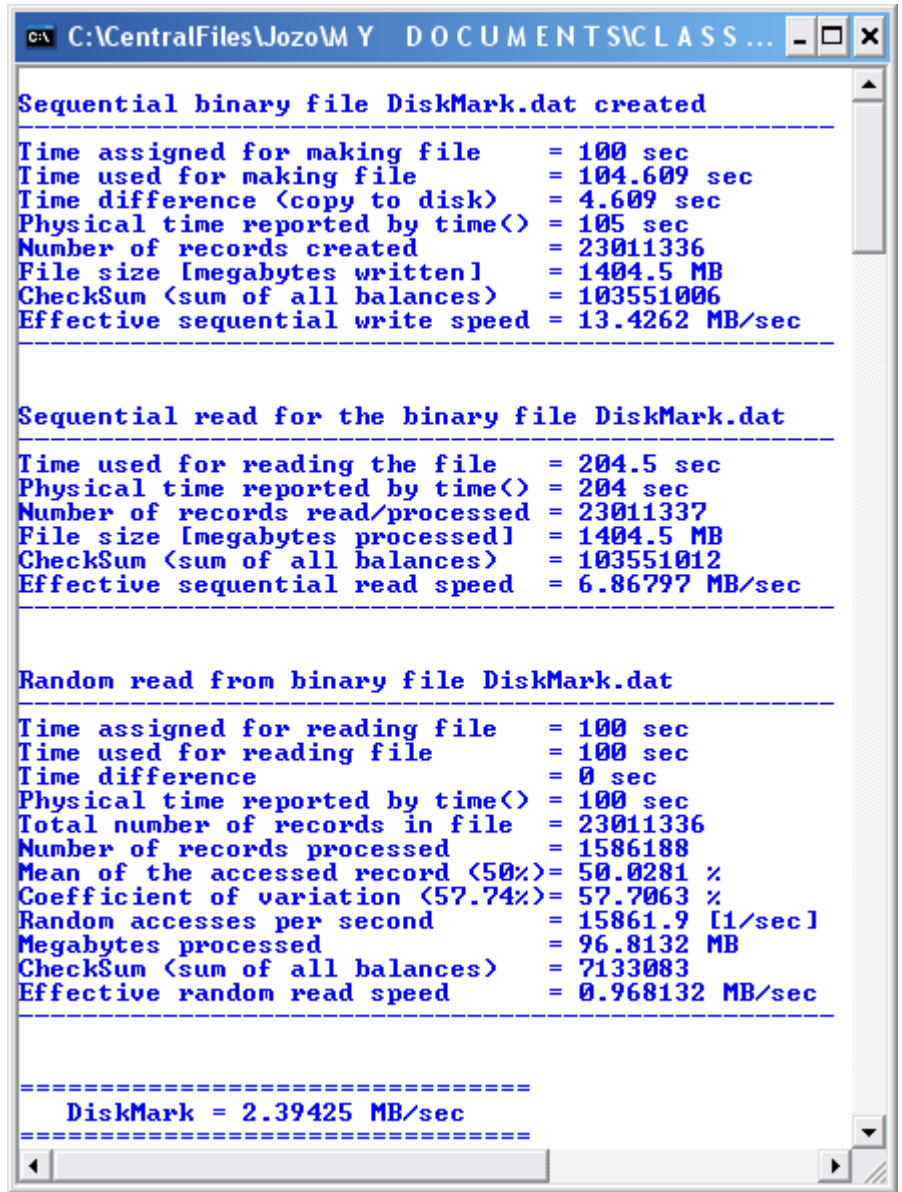
Variations caused by big files





Variations caused by big files





Conclusions

- Disk read/write benchmarks in the case of large main memory primarily depend on the file caching organization
- For small and medium files the measured performance is not significantly affected by performance of disk unit
- Disk unit performance is significant in cases of very large files or files that are not present in disk cache (e.g. virus scan)