Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 9 September 2019

Case Number: T 2083/13 - 3.5.06

Application Number: 04257684.3

Publication Number: 1544735

IPC: G06F9/46, G06F11/14

Language of the proceedings: EN

Title of invention:
Method and system of accessing a target file in a computer system with an operating system with file locking implemented at file-open time

Applicant:
Lenovo (Singapore) Pte. Ltd.

Headword:
Acessing locked files/LENOVO

Relevant legal provisions:
EPC 1973 Art. 56

Keyword:
Inventive step - (no)

Decisions cited:
T 1742/12

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern

Boards of Appeal

Chambres de recours

Boards of Appeal of the
European Patent Office
Richard-Reitzner-Allee 8
85540 Haar
GERMANY
Tel. +49 (0)89 2399-0
Fax +49 (0)89 2399-4465

Case Number: T 2083/13 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 9 September 2019

Appellant: (Applicant)	Lenovo (Singapore) Pte. Ltd. 7, Changi Business Park Central 1 Singapore 486048 (SG)
Representative:	Schweiger, Martin Schweiger & Partners Intellectual Property Law Firm Elsenheimer Strasse 1 80687 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 5 July 2013 refusing European patent application No. 04257684.3 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	M. Müller
Members:	S. Krischer
	A. Jimenez

Summary of Facts and Submissions

I. The appeal is directed against the decision of the examining division, dated 5 July 2013, to refuse application No. 04257684.3 for lack of inventive step (main request and auxiliary request 1) over D1 (US5675725) in combination with D2 (A. Baker, J. Lozano: "The Windows 2000 Device Driver Book - Second Edition", 2001, Prentice-Hall, XP2526525, ISBN 0-13-020431-5), and for added subject-matter and lack of clarity of (then) auxiliary request 2.

II. A notice of appeal was received on 5 September 2013. The appeal fee was paid on the same day. A statement of grounds of appeal was received on 26 September 2013. The main request and auxiliary request 1 were maintained. A new auxiliary request 2 was filed, replacing the former one.

III. In a communication dated 21 March 2019, the rapporteur raised an objection of lack of inventive step, using an NT-based Windows operating system as the starting point for the problem-solution approach.

IV. In a letter dated 21 May 2019, the appellant submitted its arguments.

V. In its summons to oral proceedings, the board addressed these arguments and gave further reasons as to why the claims lacked an inventive step.

VI. In a letter dated 9 August 2019, the appellant submitted further arguments and filed auxiliary request 3, as well as a request to refer the question to the Enlarged Board of Appeal whether a framework of

an operating system as such can be considered as being the closest prior art.

VII. Oral proceedings were held on 9 September 2019, during which the appellant withdrew its request to refer the question to the Enlarged Board of Appeal. At their end, the board announced its decision.

VIII. The appellant's final requests were that the decision be set aside and that a patent be granted on the basis of the claims according to:

- the main request (subject of the appealed decision), the originally filed claims;
- auxiliary request 1 (also subject of the appealed decision), filed with letter of 14 May 2013;
- auxiliary request 2, filed with the grounds of appeal;
- auxiliary request 3, filed with the letter dated 9 August 2019.

The other application documents are the same as indicated in the appealed decision.

XIII. Claim 1 of auxiliary request 3 reads as follows:

"1. A method of accessing at least one target file in a computer system with an operating system with file locking implemented at file-open time, the method comprising:

obtaining (210) a set of handles that corresponds to a set of all files that are open in the computer system;

determining (212) within the kernel of the operating system a set of file identifiers that corresponds to the set of handles;

identifying (214) from the set of file identifiers a file identifier that corresponds to the target file;

sending (216) within the kernel to the file system driver of the operating system a read request packet that corresponds to the identified file identifier; and

receiving (218) from the file system driver data that corresponds to the target file,

wherein the obtaining comprises: issuing (310) an NtQuerySystemlnformation [sic] to the kernel, wherein the NtQuerySystemlnformation asks for all information about each handle in the set of handles; and obtaining (312) from the kernel an array of SYSTEM_HANDLE_INFORMATION for each handle in the set of handles, wherein the SYSTEM_HANDLE_INFORMATION comprises a pointer to a FILE_OBJECT, wherein the FILE_OBJECT comprises file identifier information,

wherein the determining (212) comprises: for each handle in the set of handles, passing (412) a pointer to the FILE_OBJECT corresponding to the each handle to the kernel,

wherein the determining (212) further comprises: for the each handle in the set of handles, outputting (422) from within the kernel a file identifier corresponding to the passed FILE_OBJECT,

wherein the identifying (214) comprises: identifying (512) the FILE _OBJECT corresponding to the identified file identifier that corresponds to the target file,

wherein the sending comprises: passing (612) the identified FILE_OBJECT to the kernel; and requesting (614) from the kernel a reading of data from the target file corresponding to the identified FILE_OBJECT via the read request packet, wherein the read request packet comprises an Interrupt Request Packet (IRP),

wherein the requesting (614) comprises: generating (622) within the kernel an IRP corresponding to the

identified FILE_OBJECT at a certain offset and a certain length; and passing (624) the IRP to the file system driver of the operating system."

In view of the board's decision, the claim text of the other requests is immaterial, since claim 1 of the other requests contain considerably fewer features than that of auxiliary request 3.

Reasons for the Decision

1. *Summary of the invention*

1.1 The claimed invention relates to a (user level) computer process reading a part of a file (see original description, [60] and [34], in particular the last sentence; figure 2: 218) which is already open (210). Repeating these reads at different offsets until the end of the file is reached will yield a full copy of the file ([64]; not claimed). Whether the file was opened by said computer process or another one is specified neither in the claims nor in figure 2. However, the description ([2] and [4]) outlines an exemplary situation where a backup program is to save a file that has already been opened by another process. In this case, some versions of the Microsoft Windows operating system prevent the open file from being read by the backup program (while producing a "sharing violation" error). According to the grounds of appeal (page 2, last paragraph), these versions comprise Windows NT (released in July 1993) and its successors. The appellant's statement in this regard, that all versions of Microsoft Windows released after 1993 were

based on Windows NT, is inaccurate. Rather, according to https://de.wikipedia.org/wiki/Microsoft_Windows#DOS-Linie_f%C3%BCr_32-Bit-Rechner there were seven DOS-based versions of Windows released after 1993, including Windows 95 and 98. However, the board accepts that the invention is intended to be used with an NT-based Windows operating system which provides file locking at file-open time.

In the following, the abbreviation "NT" is meant to designate an "NT-based Microsoft Windows operating system".

1.2 The invention uses four programs:

- a [user level] program: e.g. a so-called "bam.exe", see [34], [53], [64]; figure 2: 210 and 214; claim 1 (of all requests): e.g. the steps of obtaining and identifying;
- a "kernel [level] program": e.g. a so-called "wam.sys", see [34], [53], [60];
- "the file system driver of the operating system": see [62] and [65]; claim 1: in the steps of sending and receiving; figure 2: 216 and 218;
- "the kernel of the operating system": see [36] and [50]; claim 1: step of determining; figure 2: 212.

1.3 Data is sent several times between the user level program and the kernel level program ([36], [50], [53], [60], [61]) in order to retrieve the NT data structure FILE_OBJECT which belongs to the file to be read (i.e. the target file). This data structure is then passed to the file system driver (via an NT IRP request), which obtains the file data and provides it to the user level program (process).

1.4 It is noted that, contrary to the appellant's allegation (see the grounds of appeal, paragraph bridging pages 10 and 11), an "Interrupt Request Packet (IRP)" (see claim 1, last step of requesting, and [61]) does not seem to exist in the NT driver models, but only an "I/O Request Packet (IRP)" (see D2, page 64, last paragraph, and e.g. https://en.wikipedia.org/wiki/I/O_request_packet). During the oral proceedings, the appellant stated that the Interrupt Request Packets are a subset of the I/O Request Packets, without however providing evidence that the term "Interrupt Request Packet" was used in the art at all, or indicating any specific characteristics that distinguished Interrupt Request Packets from I/O Request Packets. Thus, the board interprets the claimed Interrupt Request Packets as ordinary I/O Request Packets.

2. *Inventiveness*

2.1 The board agrees with the decision that the invention lacks inventive step. However, it is of the opinion that no document is needed for establishing that. As the starting point for the problem-solution approach, an NT-based Windows operating system (NT) is chosen.

2.2 In the following, only auxiliary request 3 is discussed, since it is narrower and more concrete than the other requests. Therefore, an argument showing lack of inventive step of this request also applies to the other requests.

2.3 The method as defined by claim 1 reads a part of an open target file using NT functions. This is not possible with the usual file reading routine of NT if the file was opened by another process (which is not

specified in the claims). The reason is that NT provides a rather strict file locking mechanism which forbids access to a file opened by another process.

2.4 Thus, the objective technical problem to be solved is how to read a part of an open (target) file while using NT.

2.5 In this context, it is understood that some sort of identifier of the target file (e.g. a file name) must be given.

2.6 During the oral proceedings, the appellant argued that the claimed invention had to be construed as implying the purpose of accessing the file, namely backup. This purpose was disclosed in paragraph [4], even though it related to the background art. The appellant pointed out that paragraphs [5] to [10] disclosed backup methods for open files as did two further US patents (US6415300, US8074069) which the appellant referred to but did not formally introduce. In contrast, NT did not specifically relate to backup. Therefore, NT was not a suitable starting point for the assessment of inventive step, which should rather start from any of the known backup methods.

2.7 However, the application discloses (see paragraph [4]) that it is important for many applications to read data from a file opened by a different process. Backup applications are amongst them, but are mentioned only as an example. Although other applications are not specifically identified, at least one comes to mind: in order to debug a program writing into a file, it might be important for the debugger to access that file while it is open. The board concludes that the discussion of

backup methods in the background art section does not imply that the only purpose of the claimed method is backup and that, therefore, this purpose is not an implicit feature of the method.

2.8 In passing, it is noted that even if backup was specified in the claim, this would not disqualify NT as closest prior art. Firstly, the board takes the view that the assessment of inventive step can start from essentially any document (see T 1742/12, section 9). Secondly, the board considers that providing a reading method for producing a backup in an NT system is a realistic objective technical problem, and that the skilled person addressing this problem would not be bound to modify the reading method of a known backup method, but would also assess the possibility of providing a reading method from scratch.

2.9 Finally, the board notes that the appellant itself, in its letter of 21 May 2019, proposed an objective technical problem which was not limited to backup.

2.10 Thus, the board sticks with the technical problem as formulated above.

2.11 Over NT, the specifically claimed steps for carrying out the claimed method are all new.

2.12 However, in order to solve the technical problem, the invention uses programs and a data structure *intended for this purpose*, such as the existing NT kernel routine NtQuerySystemInformation (see [36], [50]), the NT file system driver and the IRP data structure for the packet-driven I/O ([61]-[64]; see also D2, page 64 last paragraph).

2.13 The skilled person, who must be considered knowledgeable about the details of NT system routines, would know that the file system driver does not test whether the accessed file is already open and thus cannot and does not prohibit reading from open files. Thus, the skilled person would know that direct access to the file system driver is useful for solving the problem posed.

2.14 As the file system driver requires the NT data structure FILE_OBJECT to access an open file, the skilled person would obviously query the existing data structures in the kernel to retrieve this data. Likewise, the skilled person would have to obtain and provide the "offset" and "length" parameters if, and since, they are required by the file system driver (see the end of claim 1). Since some kernel data structures can only be accessed from the kernel space, the skilled person would program this kernel-specific part in a kernel level program without exercising any inventive activity.

2.15 Splitting a kernel-related task into a user level program and a kernel level program is considered by the decision (section 16) to be a well-known programming methodology. In its letter dated 21 May 2019 (page 3, paragraph 3), the appellant challenges this assumption, and in its grounds of appeal (page 7, third paragraph), it argues that even if D1 was taken to disclose such splitting, at least the "specific interaction between the user level program and the kernel level program as anticipated by claim 1" was not disclosed in D1.

2.16 The board agrees that the specifically claimed split of work between the user level and the kernel level is not

disclosed in D1 nor implied by NT. However, since, as mentioned, some kernel data structure can only be accessed from the kernel space, the skilled person would have to program this kernel-specific part in a kernel level program. The rest can be done in a user level program that is more easily accessible. This renders the claimed split between user level and kernel level programs obvious, irrespective of whether co-operation between user level and kernel level programs are, in general, a known programming concept.

2.17 In passing, the board also notes that nothing in the description hints that the application would have invented the concept of co-operating user level and kernel level programs, suggesting at least that the application was relying on this as a known concept.

2.18 The appellant states in its letter dated 21 May 2019 (page 3, paragraph 5) that there are several different ways to access these kernel data structures. For example, the computer could be rebooted into a preboot environment in order to access them.

2.19 The board disagrees. This is not possible, since these data structures (as they are disclosed in description sections [37]-[49]) are held in the main memory (RAM; during the run-time of the program), and after a reboot they do not exist anymore. Thus, they cannot be accessed after reboot.

2.20 The appellant further argues (page 3, paragraph 6) that the development of an NT is very complex wherefore it is disputed that the splitting is a well-known programming methodology for NT.

2.21 The board cannot see why the complexity of developing an operating system should have anything to do with the way of organising the interaction between user level functionality and kernel-specific functionality.

2.22 Therefore, claim 1 of auxiliary request 3 is not inventive, in violation of Article 56 EPC. It follows that broader claim 1 of each of the other requests is not inventive, either.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

I. Aperribay M. Müller

Decision electronically authenticated