Internal distribution code:

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

Datasheet for the decision of 8 March 2018

Case Number: T 2376/11 - 3.5.01

Application Number: 05735535.6

Publication Number: 1745381

IPC: G06F11/00

Language of the proceedings: EN

Title of invention:
MANAGING PROCESS STATE INFORMATION IN AN OPERATING SYSTEM ENVIRONMENT

Applicant:
Alcatel-Lucent USA Inc.

Headword:
Process state information / ALACATEL-LUCENT

Relevant legal provisions:
EPC Art. 56

Keyword:
Inventive step - pushing of process state information to the user space (no - obvious combination of known features)

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 2376/11 - 3.5.01

D E C I S I O N
of Technical Board of Appeal 3.5.01
of 8 March 2018

Appellant: (Applicant)	Alcatel-Lucent USA Inc. 600-700 Mountain Avenue Murray Hill, New Jersey 07974 (US)
Representative:	Novagraaf Technologies Bâtiment O2 2, rue Sarah Bernhardt CS90017 92665 Asnières-sur-Seine Cedex (FR)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 24 June 2011 refusing European patent application No. 05735535.6 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	W. Chandler
Members:	N. Glaser
	P. Schmitz

Summary of Facts and Submissions

I. This appeal is against the Examining Division's decision to refuse the European patent application 05735535.6. The Examining Division was of the view that the invention lacked an inventive step over a combination of D1 (US6681348) with common general knowledge or with D4 (Daniel P. Bovert & Marco Cesati, Understanding the Linux Kernel, First Edition, October 2000, pp. 1-35, 217-230).

II. With the statement setting out the grounds of appeal, dated 18 October 2011, the appellant requested that the decision under appeal be set aside and that a patent be granted based on the set of claims filed therewith.

III. After an initial communication and a reply, the Board arranged to hold oral proceedings and confirmed its provisional view in the accompanying communication. In particular, the Board tended to agree with the examining division's decision that the claimed invention did not involve an inventive step.

IV. The appellant informed the Board by letter of 5 February 2018 that he would not be attending oral proceedings and requested that the Board decide on the state of the file.

V. Independent claim 6 according to the main request reads as follows:

A method for managing process state information in an operating system environment, wherein local memory (102) used by an operating system is segregated into kernel space (106) and user space (104), the method comprising:

executing (200) a first application (110);
identifying (202) a fault in the execution of the first application; and

in response to identifying a fault in the execution of the first application, pushing (204) process state information (122) from the kernel space of the local memory directly to a core dump application (112) in the user space of the local memory;

generating a core file (124) in the user space of the local memory from the process state information that is pushed directly to the user space, wherein the core file is generated at the user space instead of at the kernel space; and

forwarding (133) the core file from the user space of the local memory to a remote system.

VI. The appellant's arguments can be summarized as follows :

D1 as closest prior art shows a conventional crash or core dump process. It does not disclose the features of (a) pushing process state information from the kernel space of the local memory directly to the user space of the local memory, (b) a core dump application using the user space of the local memory and configured to generate a core file from the [pushed] process state information, and (c) wherein the core file is generated at the user space *instead of* at the kernel space.

These features in combination solve the technical problem of limited local permanent storage capacity for storing an initial kernel level crash/core dump file on the local machine by handling process state information

at the user level instead of at the kernel level. The invention pushes process information to the user level and generates the core file at the user level.

Reasons for the Decision

1. The invention concerns a system and method for managing process state information in an operating environment when the execution of an application faults and when process state information is dumped to storage.

2. D1 is considered to be the closest prior art. D1 discloses the generation of mini dump or summary files which are better suited for transmission over a communication link for further analysis or for storage to a portable memory device, column 1, lines 30 to 63, column 2, lines 52 to 55, column 5, lines 1 to 4, and column 8, lines 36ff. More importantly, D1 is not limited to the generation of a (complete) system crash dump file, but discloses the generation of an application crash dump file as well, column 8, lines 25ff. Both types of crash dump files are said to be very big in memory size and it is often impossible to predict the memory necessary to store it, because the amount of kernel-mode memory allocated by the operating system varies. Therefore the problem of (limited) memory storage is addressed in D1. Furthermore, D1 is not limited to a kernel crash dump process 26 in kernel mode and a user mode crash dump process 240 in user mode, but both processes can be combined, column 10, lines 21 to 28, to achieve flexibility.

3. Accordingly, D1 discloses a method for managing process state information in an operating system environment, wherein local memory used by an operating system is

segregated into kernel space and user space (D1: Figure 1, items 14 and 16; column 4, lines 29-31), executing a first application (D1: Figure 1, item 18; column 4, lines 31 to 35), identifying a fault in the execution of the first application (D1: column 8, lines 30-57), in response to identifying a fault in the execution of the first application (D1: column 8, lines 30-57), generating a core file in the user space of the local memory from the process state information (D1: Figure 7, item 220; column 8, lines 40-50; Figure 8, item 240; column 8, line 58, to column 9, line 23), forwarding the core file from the user space of the local memory to a remote system (D1: column 2,lines 59-61).

4. Claim 6 therefore differs from D1 by the following features :

(a) pushing process state information from the kernel space of the local memory directly to a core dump application in the user space of the local memory;

(b) generating a core file in the user space of the local memory from the process state information that is pushed directly to the user space,

(c) wherein the core file is generated at the user space instead of at the kernel space.

5. The Board is not convinced that feature (c), added by amendment during the appeal procedure, adds a further technical difference over D1. Features (a) and (b) imply that the core file is generated at the user space and not at the kernel space, because the core dump application is using the user space of the local memory to which the process state information from the kernel level is pushed.

6. The objective technical problem deriving from the differentiating features (a) to (c) may be regarded as that set out by the examining division, namely to provide_a more complete picture of the overall state of the operating system at the user space, following an execution fault.

7. Facing the above objective technical problem, the person skilled in the art would employ well-known data transfer functions for the transfer of data between user and kernel space, see for example, D4, page 225, Table 8-1, to realise a transfer of process state information from the kernel space to the user space in order to create a core file which includes kernel space data as well. The Board judges that the adaption of the stand-alone extraction tool 240 or the user mode crash dump process 220 would be straight-forward.

8. Claim 6 therefore lacks an inventive step over a combination of D1 with D4. The Board further notes that D1 suggests a combination of core files comprising kernel and application information, see D1, column 8, lines 26 to 57, and column 10, lines 21 to 28.

9. Contrary to the appellant's argument that D1 would disclose only two options of crash dump files, the Board considers D1 to disclose three options of crash dump files: a complete memory crash dump file, a kernel mode crash dump file and a user mode crash dump file, see D1, column 1, lines 45 to 48, column 2, lines 42 to 46, and column 4, lines 60 to 65. In particular, column 4, lines 58 to 67, explains that a complete memory dump file contains all of the physical memory present on the system, that is, the user mode and the kernel mode portion of the physical memory component. On the other hand, the kernel mode crash dump file contains only the

kernel-mode read/write pages and no pages belonging to user processes. A user mode crash dump file is a crash dump file of an application at the time of a crash, see D1, column 8, lines 32 to 39.

D1 therefore explicitly discloses a crash dump file containing process state information of user mode, kernel mode or both together.

10. The appellant further argued that the generation of a complete memory crash dump file would be a complete solution to the objective technical problem of having a more complete picture of the overall state of the operating system, following an execution fault.

The appellant's argument is not convincing, because D1 explicitly discloses generating a complete memory dump file or a system crash dump file that represents the state of the entire physical memory, see column 8, lines 25 to 29.

11. The appellant further argued that the application does not address the content of the (kernel mode) core dump file, but the way in which it is generated, namely via a more flexible user process at user level instead of a more inflexible kernel process at kernel level. The appellant considers that the technical objective problem is therefore to provide an increased flexibility with respect to the storage of process state information from kernel space.

These arguments are not convincing, because no detailed explanation has been given by the appellant why a user process should be more flexible in the generation and management of a core file compared to a kernel process. In both situations the core dump process appears to

collect process state information and to store it in the form of a crash dump file.

The choice of creating one or the other crash dump file depends on the location of the error leading to the crash, that is, whether the error occurred in the kernel mode portion or in the user mode portion, see D1, column 4, lines 15 to 19, and column 8, lines 26 to 39. According to column 4, lines 47ff., the kernel mode crash dump file 30 is generated by a crash dump process 26 residing in the kernel mode portion 16 of a memory component, and according to column 8, lines 47ff., the user mode crash dump 230 is generated by a crash dump process 220.

12. Contrary to the appellant's statement that D4 would disclose only unspecific and generic data transfer functions, the functions listed in Table 8-1 have a specific role which is to access the process address space in the kernel mode for implementing the process/ kernel model described in Section 1.6.1.

13. Even if accepted, the objective technical problem suggested by the appellant to overcome insufficient local memory for storing the kernel crash/core dump file, does not lead to an inventive step.

14. This objective problem assumes that the user space of the local memory has unlimited storage capacity, while the kernel space is limited. However, the limitation of the local storage capacity, as mentioned in paragraph [5] of the application description, appears to apply to both, the kernel space as well as to the user space, because paragraph [19] of the application description lists various methods to address memory limitations in the user space. Such methods are a compression of the

core file, an extraction of only certain information from it or a forward of the file to another system which has permanent storage capacity available.

15. The purpose of the crash dump file of the present application is, however, not different than in D1, namely to collect state information about a system and process(es) when they fail. This information is then used for debugging and failure analysis, see D1, column 1, second paragraph, and is therefore expected to be as complete as possible. It is a matter of limited obvious choices, whether to store the collected process state information in the form of a crash dump file at the user or kernel space.

16. Completeness leads to files of nearly "astronomical" size, see D1, column 1, lines 35-37, in particular for full crash dump files, and to storage problems, because of a limited memory capacity, but also to transmission problems to other computers, D1, column 8, lines 35ff. This problem has also been recognised in the present application, see paragraph 19, and was said to have motivated the claimed invention.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

T. Buschek W. Chandler

Decision electronically authenticated