Internal distribution code:

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

Datasheet for the decision of 1 July 2024

Case Number: T 1956/21 - 3.5.06

Application Number: 16154359.0

Publication Number: 3056991

IPC: G06F9/48

Language of the proceedings: EN

Title of invention:
APPARATUS AND METHOD FOR MANAGING A PLURALITY OF THREADS IN AN OPERATING SYSTEM

Applicant:
Honeywell International Inc.

Headword:
Thread management/HONEYWELL

Relevant legal provisions:
EPC Art. 84

Keyword:
Claims - clarity (no)

Decisions cited:

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 1956/21 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 1 July 2024

Appellant: (Applicant)	Honeywell International Inc. 115 Tabor Road M/S 4D3 P.O. Box 377 Morris Plains, NJ 07950 (US)
Representative:	Houghton, Mark Phillip Patent Outsourcing Limited 1 King Street Bakewell, Derbyshire DE45 1DZ (GB)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 26 April 2021 refusing European patent application No. 16154359.0 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	M. Müller
Members:	G. Zucka
	K. Kerber-Zubrzycka

Summary of Facts and Submissions

I. The appeal is against the decision by the examining division, dispatched with reasons on 26 April 2021, to refuse European patent application 16154359.0, on the basis that the main request did not satisfy the requirements of Article 56 EPC and both auxiliary requests did not satisfy the requirements of Article 123(2) EPC. The following document was cited during the first instance proceedings:

D1: US 2014/282507 A1.

II. A notice of appeal was received on 25 June 2021, the appeal fee being paid on the same day. A statement of grounds of appeal was received on 16 August 2021.

III. The appellant requested that the decision of the examining division to refuse the application be set aside and a patent be granted on the basis of claims 1 to 15 of the main request that was the object of the refusal, re-filed with the statement of grounds of appeal on 16 August 2021, or of auxiliary request 1 or 2, filed with the statement of grounds of appeal on 16 August 2021. The appellant made a conditional request for oral proceedings.

IV. The board issued a summons to oral proceedings. In an annex to the summons, the board set out its preliminary opinion on the appeal, according to which the appealed decision should be maintained.

V. On 31 May 2024, the appellant filed claims 1 to 8 for an additional auxiliary request 3, and claims 1 to 6 for an additional auxiliary request 4.

VI. On 18 June 2024 (with a letter dated 6 February 2024), the appellant withdrew its request for oral proceedings, which were subsequently cancelled.

VII. The appellant requests that the decision under appeal be set aside and a patent be granted on the basis of the claims of any of the five requests mentioned above, and description pages 1 to 19 and drawing sheets 1 to 4 filed with the statement of grounds of appeal on 16 August 2021.

VIII. Claim 1 of the main request reads as follows:

"A method for managing a plurality of threads (230), the method comprising:

providing (405) an environment associated with an operating system (240) to execute one or more threads of the plurality of threads, the environment comprising a plurality of virtual priorities (210) known to the threads and a plurality of actual priorities (220) for executing threads known to the operating system, wherein each of the plurality of threads selects its own virtual priority, wherein the plurality of virtual priorities comprises a broader range of values from a highest priority to a lowest priority than the plurality of actual priorities comprise a range of values from a highest priority to lowest priority;

associating (415), by a thread selection algorithm (230x), an actual priority value to one of the plurality of threads based on its virtual priority value; and

executing (420), by the thread selection algorithm (230x), the one of the plurality of threads associated with the actual priority value."

IX. Compared to the main request, claim 1 of auxiliary request 1 includes the following method step between the steps of "associating" and "executing":

"selecting a thread using the thread selection algorithm whenever at least one of:
the thread in the actual priority blocks at least one of semaphore, timeout, and input or output function I/O;
the thread in the actual priority calls a Change_Priority function;
the thread in the actual priority calls a terminate function;
the thread in the actual priority calls a Establish_Task function; and
the thread in the actual priority calls an Enable_Interrupts or Enable_Task_Scheduling function".

X. Compared to the main request, claim 1 of auxiliary request 2 more precisely defines the "plurality of virtual priorities" in that it comprises "a range of values from a highest priority 255 to a lowest priority 0 and the "plurality of actual priorities" in that it comprises "a range of values from a highest priority 4 to lowest priority 1".

XI. Claim 9 of the main and the first two auxiliary requests relates to an apparatus having apparatus features corresponding to the method features of claim 1 of the respective request.

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

"A method for managing a plurality of threads (230), the method comprising:

providing (405) an environment which is part an operating system (240) to execute one or more threads of the plurality of threads, and configured to control the priorities of threads the environment comprising a plurality of virtual priorities (210) and a plurality of actual priorities (220);

wherein priorities 220 are priorities known to the operating system and virtual priorities (210) are priorities known to the threads (230);

wherein each of the plurality of threads selects a virtual priority using a thread selection algorithm (230x) from the plurality of virtual priorities, at initialization and wherein the plurality of virtual priorities comprises a broader range of values from a highest priority to a lowest priority than the plurality of actual priorities comprise a range of values from a highest priority to lowest priority;

associating (415), by the thread selection algorithm (230x), an actual priority of the plurality of actual priorities and to one of the plurality of threads, so as to schedule the thread for execution;

the association being based on the value assigned to the plurality of virtual priorities assigned to the plurality of threads; and

executing (420), by the thread selection algorithm (230x), the one of the plurality of threads associated with the actual priority, the virtual priority value changing dynamically by the thread during runtime."

XIII. Compared to auxiliary request 1, the "selecting" step in claim 1 of auxiliary request 4 reads as follows:

"selecting a thread for execution using the thread selection algorithm whenever at least one of:

the thread in the actual priority blocks is an input or output function I/O;

the thread in the actual priority calls an Enable_Interrupts or Enable_Task_Scheduling function".

Reasons for the Decision

1. *The invention*

Following the wording of claim 1, the application relates to an operating system executing one or more of a plurality of threads. The "actual" priority value of a thread determines whether it will be executed at a given moment and is somehow based on its "virtual" priority value, which is set by the thread itself. The range of virtual priority values is broader than that of the actual priority values.

From the "background" section on page 1 of the description, it would seem, although not stated as such, that the aim of the application is to provide a broader (even "unlimited"; see par. [0027]) range of priorities than are defined by the operating system, as well as a strict enforcement of priorities, both of which some applications are said to need.

2. *Clarity (Article 84 EPC)*

2.1 In claim 1 of the main request, it is not clear what is the nature of the "environment associated with" an operating system, except that it executes threads in some unspecified manner, and that it contains some "virtual" and "actual" priorities, which presumably are

numerical values stored in some programming variables or in some physical storage.

2.2 It is not clear how the "actual priority values" determine what happens with the various threads. The claim merely specifies as a final step that some thread associated with some actual priority value is executed. Nothing is said about what happens with other threads having the same or a different priority value.

Moreover, it is specified in the final step that "the" thread associated with "the" actual priority value is executed. "The" actual priority value apparently refers to the actual priority value which, in the previous step, has just been associated with one of the threads in an undefined manner. According to the claim, this actual priority value can be any of the available actual priority values. In particular it need not be the highest actual priority value. Nonetheless, this arbitrary actual priority value seems to be decisive for the decision which thread to execute next. No scheduling strategy emerges.

Since the claim is supposed to relate to a scheduling method which is based on priorities, such apparent lack of actual scheduling would make no sense to the reader.

It is further not clear how the actual priority value associated to a thread is selected based on that thread's virtual priority. A priori, the mapping could be arbitrary. There need not even be a fixed "mapping" at all. But even if, say, one were to assume that per actual priority value there are (say) 256 virtual priority values, and the associated actual priority value would be the virtual priority value divided by 256 and rounded up or down, the role of the virtual

priority values would be unclear. Effectively, the fact that more virtual priority values are available than there are actual priority values is lost. Or in other words, the claim does not set out how the larger range of virtual priority values actually affects the scheduling algorithm, e.g. by making it more "flexible".

The bottom line is that in particular the role of the virtual priority values - apparently *the* central contribution of the invention - for the scheduling algorithm is unclear.

2.3 It is also noted that the language used to specify the last step suggests that there might be only one thread with "the" specific actual priority value just chosen.

2.4 Claim 1 and for similar reasons claim 9 of the main request are therefore unclear (Article 84 EPC).

2.5 In claim 9, it is additionally unclear (Article 84 EPC) what is the significance of the running thread's virtual priority "changing dynamically". This feature seems to imply that a thread can change its own virtual priority during runtime in an arbitrary manner and thereby affect, indirectly, the actual priority value associated with it by the thread selection algorithm. The overall effect of this operation is unclear. Moreover, as the claim leaves open how the actual priority value of a thread is selected "based on" its virtual priority, the function of the virtual priorities for the claimed scheduling algorithm remains unclear.

2.6 The same objections apply to claims 1 and 9 of auxiliary requests 1 and 2.

Regarding auxiliary request 1, the conditions which have been added to claims 1 and 9 indicate when a thread should be selected using the thread selection algorithm, but they do not further specify how the actual selection is made. Furthermore, reference is made to a number of "functions" which are given a name but are not defined.

Regarding auxiliary request 2, the specific ranges of actual and virtual priority values cannot clarify the scheduling algorithm itself.

2.7 In claim 1 of auxiliary request 3, the expression "an environment associated with an operating system" has been replaced by "an environment which is part [of] an operating system". This however still leaves open how the term "environment" is to be understood.

The claim now specifies that "priorities 220 are known to the operating system and virtual priorities (210) are priorities known to the threads (230)". Also this, however, does not further clarify the scheduling algorithm itself.

The other clarity objections formulated for the main request remain valid also for auxiliary request 3.

2.8 All the clarity objections formulated for claim 1 of the main request remain valid for claim 1 of auxiliary request 4, as the latter contains fewer features than the former.

2.9 The appellant provides some explanation in its response received on 31 May 2024 as to how it considers that some of the expressions in the claims should be understood ("a reference point to at least what is

intended to be claimed", see page 3, last paragraph), and how "the skilled person" would understand the claimed invention "when referring to the description to resolve ambiguity Art. 69(1) EPC". However, in general such explanation - or mere reference - cannot serve as a substitute for a clear claim language, which is required by Article 84 EPC.

Moreover, the appellant's explanations are insufficient to clarify the function even of the intended scheduling algorithm and, especially, the function of the virtual priorities in this context.

2.10 The board concludes that none of the requests satisfy the requirements of Article 84 EPC.

3. The further observations made on inventive step in the board's preliminary opinion are therefore immaterial for this decision.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

L. Stridde M. Müller

Decision electronically authenticated