TECHNISCHE FAKULTÄT

Department Informatik

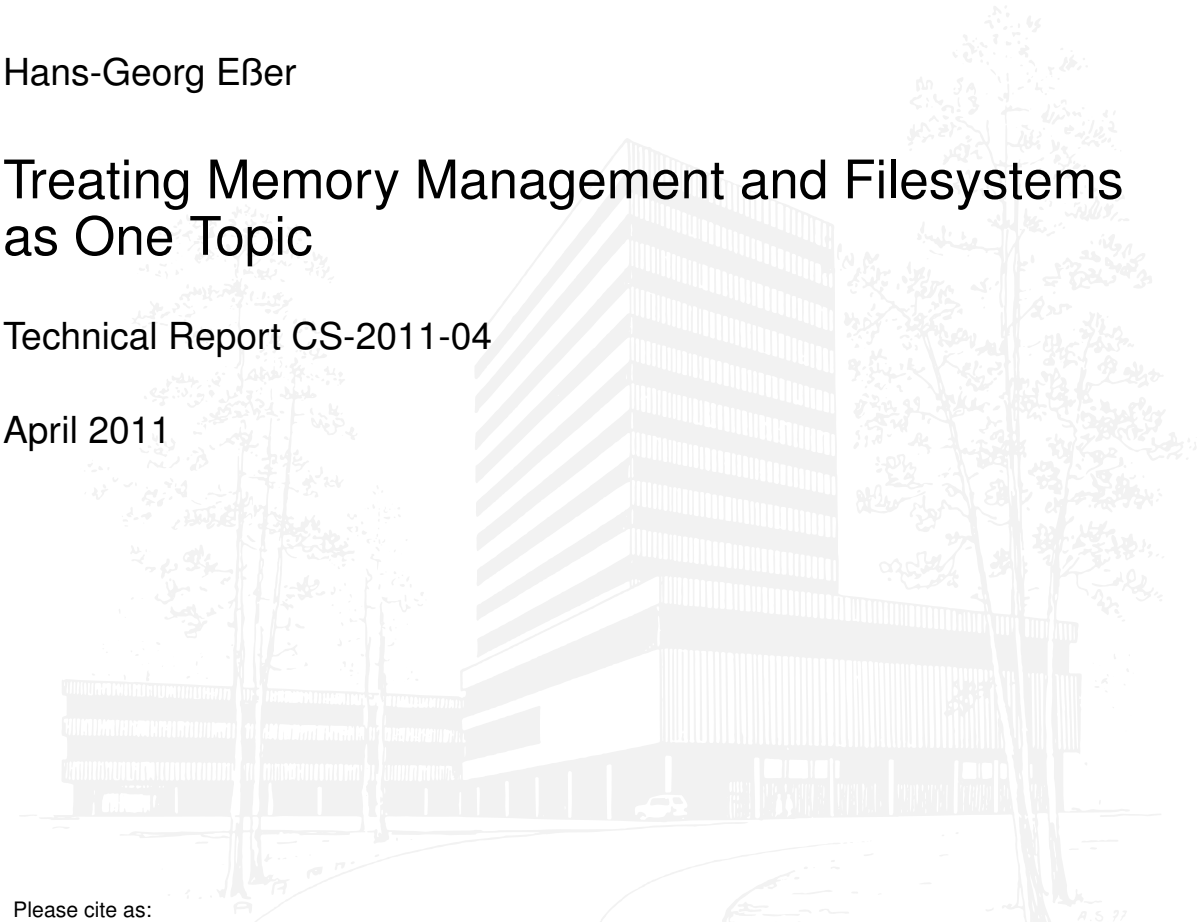
Technical Reports / ISSN 2191-5008

Hans-Georg Eßer

Treating Memory Management and Filesystems as One Topic

Technical Report CS-2011-04

April 2011



Please cite as:
Hans-Georg Eßer, “Treating Memory Management and Filesystems as One Topic,” University of Erlangen, Dept. of Computer Science, Technical Reports, CS-2011-04, April 2011.

Friedrich-Alexander-Universität Erlangen-Nürnberg

Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Informatik

Martensstr. 3 · 91058 Erlangen · Germany

www.informatik.uni-erlangen.de

Treating Memory Management and Filesystems as One Topic

Hans-Georg Eßer
IT Security Infrastructures
Dept. of Computer Science, University of Erlangen, Germany
h.g.esser@informatik.uni-erlangen.de

***Abstract*—Teaching memory management aligned with filesystems in an Operating Systems course instead of treating them as separate topics can increase students' understanding and improve their grades in end-of-term examinations. In a survey they also state that they like this method. This article describes modifications made to a classic course and the results gained through exam and survey evaluations.**

***Index Terms*—Operating Systems, Didactics, Memory Management, Filesystems**

I. Introduction

When we gave an Introduction to Operating Systems (OS) course in summer 2009, we tested a new way of presenting the two topics Memory Management (MM) and Filesystems (FS) and evaluated the effects on the students.

Most classical OS courses contain a superset of these topics:

- Processes (and Threads)
- Scheduling
- Synchronization and Deadlocks
- Memory Management
- Filesystems

Threads are not always part of an introductory OS course, since almost all relevant principles can be understood by exclusively looking at processes. Sometimes the filesystems are not treated within the first course and are left for a second course that deals with further aspects of OS. However, all courses known to us which do contain both memory management (MM) and filesystems (FS) keep these two topics separate.

We felt that there was a certain redundancy in treating MM and FS concepts separately. For example, in simple partitioning / allocation schemes, such as fixed size partitioning, where each process is given a fixed amount of memory or a file can use a fixed amount of disk space, students have to see identical concepts twice. Other examples are

- contiguous allocation vs. non-contiguous allocation,
- internal and external fragmentation,
- using bitmaps to keep track of free areas (on disk or in memory), and
- indirection (for keeping a list of blocks used by a file) vs. paging with split page tables.

The goal was to test whether a combined treatment of MM and FS concepts could improve the students' understanding while repurposing redundant lecture time (for e. g. explaining fragmentation in both contexts). Thus we created a modified OS course for the summer term 2009 that was based on the same course held in summer 2008, but with MM and FS combined into one chapter.

We evaluated the results of these changes with two methods:

- Class results in the end-of-term exam were compared, for this purpose each question in the tests was classified as MM related, FS related, or other, and students' individual marks on each question were recorded separately. The percentage P_{FM} of gained marks in FS/MM questions was then compared with the overall percentage P_T of gained marks. We observed an increase from 87.9 % to 97.8 % in the ratio P_{FM}/P_T.
- Students were given a questionnaire at the end of the term, and they were asked their opinions about the combined treatment. The results gained through this questionnaire are positive, too, but they are weaker in that the students could not base their assessments on knowing both types of teaching OS concepts, but only the new combined approach.

An extended abstract of this report appears in the ITiCSE '11 proceedings [5].

A. Outline

Section II discusses some topics which are relevant for both FS and MM and motivates the combined treatment.

In section III some other alternative approaches to teaching OS principles are presented.

Sections IV and V present the modified course and the results of the two evaluations.

Finally, section VI concludes the article with a suggestion for experimenting with combined MM/FS treatment in other OS courses.

II. Common Topics in Filesystems and Memory Management

We now look at some of the common topics in more detail. Readers familiar with OS principles may decide to skip to the next section, others may want to refer to an OS textbook such as Stallings [12] to which we will give references.

A. Allocation

Since the course is an introduction to OS, simple concepts are presented in the beginning. For both MM and FS partitioning deals with the questions

- How to split a disk so that several files can be written to the disk?
- How to split the main memory (RAM) so that several processes (and the OS itself) can reside in memory?

The simplest approach is called fixed-size partitioning and splits e. g. 1 GiB[1] of RAM into chunks of 128 MiB allowing for up to eight processes to be placed in RAM (actually only seven or less since part of the RAM will be reserved by the OS). In a similar way a disk of size 1 GiB can be partitioned into chunks of 128 MiB, letting the system store up to eight files on the disk.

In both cases this approach has the same problems (inflexible: no more than eight processes/files, size limitation of 128 MiB per process/file), and in the next step a modification called variable-size partitioning is introduced. Variability can mean that there is still a fixed partitioning of disk or memory (but with varying partition sizes) or that any pre-partitioning is given up completely in favor of dynamically partitioning as the system runs.

Ultimately this series of incremental improvements to the initial fixed-size partitioning leads from contiguous to non-contiguous allocation methods, delivering different results for MM and FS, e. g. paging which has no FS equivalent and file allocation tables with indirection blocks which has no MM equivalent. Note however that split page tables show some relatedness to file access via multiple indirection blocks, which uses similar methods to solve the lookup problem. (Where in memory does the n-th page of a process reside? Where on the disk does the n-th block of a file reside?)

Looking at MM and FS at the same time can yield interesting questions, such as: When paging solves the out-of-memory problem by storing memory content on disk, is there a similar method to solve the out-of-disk-space problem?[2]

See chapters 7.2 and 12.6 of Stallings [12] for further information.

B. The Locality Principle

For both MM and FS the locality principle states that access sequences often have a locality property, consider e. g. a loop through an array of integers stored in RAM or the complete sequential reading of a file. When creating a MM subsystem or a filesystem it makes sense to do it in such a way that performance profits from locality. In the case of memory, the translation look-aside buffer (TLB) and the CPU's internal cache(s) can increase access to memory locations in the same page; for filesystems caching reads from the disk or writes to the disk increases speed when the system caches and buffers blocks after reading and before writing.

For a detailed description of locality and the TLB, see chapter 8.1 of Stallings [12].

C. Internal and External Fragmentation

Internal fragmentation is a waste of resources: memory or disk space has been allocated to a process or a file and remains unused. It occurs maximally e. g. in paging when there is some page size N and the process requires $kN+1$ bytes of RAM; it occurs in file storage when the least allocatable (block) size is N and a file has size $kN+1$.

External fragmentation is a result of contiguous allocation in which repeated allocations and de-allocations lead to the existence of small holes between allocated areas which are too small to be useful for a process or a file. Note that this does not exactly match the fragmentation

[1] In this paper IEC prefixes are used: 1 KiB = 2^{10} bytes, 1 MiB = 2^{20} bytes, and 1 GiB = 2^{30} bytes.

[2] Actually there is, if the memory hierarchy is extended with a robot-operated magnetic tape library, though this is more related to swapping than to paging. The technology is typically referred to as Hierarchical Storage Management.

that occurs in modern filesystems where it means non-contiguous allocation with portions of a file spread all over the disk, though it is related since in general a non-contiguous filesystem tries to store files contiguously; defragmenters enforce this.

When fragmentation is presented for MM and FS, details of the internal and external variants need only be explained once; the concepts translate directly from one topic to the other. Even the idea of a disk fragmenter can partially translate to memory, think of garbage collection with compactification in interpreter languages.

See chapter 7.2 of Stallings [12] for further information.

D. Topics without Equivalents

There are a few topics in both the MM and FS parts which have no equivalent in the other area and thus will not profit directly from the combined approach. They can be inserted in the course where they fit naturally. Examples are:

- Replacement Strategies: For paging there are various page replacement strategies whose goal it is to reduce the overall number of page faults. (A page fault occurs when a memory page was paged out to disk and a memory position within this page is accessed: the page then has to be reloaded from the disk and the process has to be blocked until this action has completed.) Nothing similar exists for filesystems, unless a Hierarchical Storage Management is in use.
- Journaling: Modern filesystems use journaling to store information about metadata changes (or in some cases data changes) before actually performing those changes in order to make recovery from an unexpected system failure faster; they remove the need for a filesystem check to scan the whole disk. There is no related procedure for dealing with memory.

E. Making the Affinity Explicit

In a traditional OS course students will also note that some concepts from MM reappear in FS (or the other way round), while some concepts do not. However, the modifications suggested and tested by us make this affinity explicit. The similarity of MM and FS in some areas becomes obvious and lets students focus on overall concepts instead of details (while not neglecting the details but letting students put them into the whole picture more easily).

This should make students more capable of transferring knowledge from one area to another. There are other topics in Computer Science where solutions to problems exist which are similar to those found in OS. For example a student attempting to learn about databases may find it easier to understand the atomicity property of a transaction if he/she previously dealt with synchronization issues in an OS course.

III. Related Work

Most research in the domain of didactics in operating systems focuses on lab exercises, some attempts have been based on educational theory, e. g. on constructivism. The problem that has been receiving the most attention is: How can the theoretical lectures on OS concepts enable students to actually implement or modify the implementation of operating systems? To this end several approaches have been tested, since educators have several choices:

- It is possible to show the creation of a complete yet simple system from scratch, as was done by Tanenbaum [13] who includes the full source code of the Minix operating system in his Minix book and refers to code in the text. Hartley [7] used Minix in an OS lab class that followed a theoretical OS course.
- Another option is to start with a working limited system and let the students extend its functionality by e. g. including a more sophisticated memory management subsystem or improving the filesystem [1].
- Some have suggested to use a real-world system, such as Linux or a BSD variant [2], of which full source code is available, and focus on understanding aspects of its implementation.
- All of these work with systems that execute on real hardware. An alternative is to use a system that is intended to run on a virtual machine, where both the OS and the virtual hardware were designed for teaching purposes and thus do not have to deal with real-world problems. MMIX [8], SOsim [9], and ULIX [6] follow this approach.

A lower-level alternative is to only teach students to use the functions (system calls) provided by the OS, e. g. by writing multi-threaded programs which use synchronization primitives available in the OS [14] or writing a `ps`-like tool by extracting process information from the kernel [11].

However, none of the recent publications have questioned the traditional treatment of memory management

and filesystem concepts as separate entities. From a didactical point of view it is valid to ask which concepts should be taught in which order. For example, van Merriënboer et al. [10] suggest "whole task learning" (keeping complex problems intact instead of breaking them down into small and well-understandable but useless sub-problems) and the "variability principle" (using a concept, procedure, etc. in various settings) in his research on 4C/ID (Four Components Instructional Design). In the context of operating systems these lead to attempting to teach OS principles in a way such that the overall connectedness of all separate OS topics is apparent and recurring concepts are shown in several areas.

IV. The Modified Course

Modifications were made on a course held in summer 2008. The original course had the following structure:

1) Processes and Threads,
2) Interrupts,
3) Scheduling,
4) Synchronization and Deadlocks,
5) Filesystems,
6) Memory Management.

For the new course all parts except FS and MM were left unchanged, while the two topics were combined into one larger chapter. The following list is an outline of the FS/MM chapter, lecture notes on this chapter which were available to the students, can be found online [3]. In addition our slides are online, too, though only in German language [4]. The combined FS/MM part discussed the following topics:

1) Introduction / Overview
   a) Tasks of Filesystems
   b) Tasks of Memory Management
2) Contiguous / Non-Contiguous Allocation
3) Internal and External Fragmentation
4) Contiguous Allocation
   a) Dynamic Partitioning
   b) Handling Free Space: Linked Lists, Bit Maps
   c) Allocation: First-Fit, Best-Fit, Worst-Fit, Quick-Fit, Buddy System
   d) MM: Code Relocation, Memory Protection
5) Non-Contiguous Allocation
   a) Blocks (FS) and Pages (MM)
   b) MM: Segmentation
   c) FS: Indirection with Multi-Layer Index Blocks
   d) FS: Unix Filesystems, Linux VFS, Ext3 FS
   e) MM: Virtual Memory (Paging)
      i) Translation Look-Aside Buffer
      ii) Inverted Page Tables
      iii) Multi-Layer Paging
      iv) Page Faults, Page Replacement Strategies
   f) Locality Principle
6) FS: Swapping

While the 2008 course consisted of 33 lessons (à 45 minutes) of which six lessons dealt with MM and five lessons with FS, the new (2009) course was 34 lessons long and treated MM and FS in 11 lessons.

V. Evaluation

This section presents results from students' exams and their feedback via a questionnaire as well as an interpretation of these results.

A. Exam Results

In both terms the exams allowed students a selection of questions to work on. With all questions worth 130 points, 95 points already led to the best available grade (1.0 in the German scale).

- In the old course (summer 2008) 27 students took the final exam. 130 points were divided into 34 points (26.2 %) for MM questions, 24 points (18.5 %) for FS questions, and 72 points (55.4 %) for other topics.
- One year later 16 students took the final exam. A total of 130 points were divided into 29 points (22.3 %) for MM questions, 27 points (20.8 %) for FS questions, and 74 points (56.9 %) for other topics.

Thus the distribution of the topics was similar in both exams. In both courses students had the opportunity to participate in a trial exam with tasks similar to the ones in the final exam. Also students from both years were able to find and download old exam questions from the Internet if they cared to search for them. Thus, means of exam preparation were very similar for both groups.

Table I shows in its upper part how many of the available marks were gained by the students on average. Each exam question was classified as either (a) Memory Management (MM) related, (b) Filesystems (FS) related, or (c) not related to either. In cases where questions required knowledge belonging to more than one class, marks for "subquestions" were recorded separately.

Since it is not helpful to compare students' successes from two different academic years, it makes sense to look at relative success: we compared how well students handled MM and FS questions with their overall performance by calculating quotients such as P_M/P_T (where

TABLE I
COMPARISON OF STUDENTS' EXAM RESULTS.

(I a) Values before normalizing

		Old	New
P_M	Memory Management (MM)	48.20%	52.59%
P_F	Filesystems (FS)	58.80%	72.92%
P_{FM}	FS+MM	52.59%	62.39%
P_T	overall	59.83%	63.80%

(I b) Values after normalizing *)

		Old	New	Change
P_M/P_T	MM	80.56%	82.43%	+ 2.32%
P_F/P_T	FS	98.28%	114.29%	+16.30%
P_{FM}/P_T	MM+FS	87.90%	97.79%	+11.25%
P_T/P_T	overall	100.00%	100.00%	—

*) Percentages in Table I b were derived from those in Table I a by dividing each value by the same column's overall value.

TABLE II
EXAM RESULTS IN FILESYSTEM-LESS COURSE.

Other Course	Absolute	Normalized
MM	70.00%	109.38%
FS	0.00%	0.00%
overall	64.00%	100.00%

TABLE III
GENERAL ANSWERS FROM QUESTIONNAIRES.

Course	Old	New
Lectures were interesting	1.6	1.2
I liked coming to the lectures	1.6	1.4
Lectures were well-structured	1.2	1.1
I would recommend this course	1.3	1.0
Overall grade for this course	1.4	1.1
Grade for lecturer's didactics	1.6	1.4

P_M is the average percentage of MM points gained in the exams and P_T is the average overall percentage of points). The lower part of Table I shows the results: in the new course all these relative values are higher than in the old one.

In addition the exam results of 2009 were also compared with those of another Operating Systems course held at the same university and in the same term by Prof. Vogt, a different lecturer. One of the major differences in that course was that filesystems were not treated at all, since they were left for an "Operating Systems II" course in the following term. 22 students took that exam, on average achieving 64.0 out of 100 points (64 %). Of those 100 points, 24 were for memory management questions, and the test takers received an average of 16.8 of those 24 points (70 %) (cf. Table II).

B. Student Questionnaire

In the last lessons of both lectures end-of-term evaluation forms were given to the students, with the 2009 form containing extra questions which dealt spefically with the combined treatment of FS and MM topics. Table IV on the next page shows the results: nine of the ten evaluation participants (90 %) stated that the combined treatment made sense (50 % agreed completely, 40 % agreed with this statement), and 80 % said that the frequent changes between FS and MM did not cause confusions (10 % said it was confusing). All of the ten participants agreed with the statement "The combination made it easy to understand that many concepts from one topic translate to the other topic".

However, the question whether more topics should be combined in a similar fashion in future courses was answered more sceptically – 30 % approved, 20 % disapproved, and 50 % were neutral or had no opinion.

Table III shows answers to some general questions about the course. Students could give marks on a 1–5 scale with 1 meaning "total agreement" and 5 meaning "total disagreement"; for the last two questions grades could vary between 1 = very good, 2 = good, 3 = satisfactory, 4 = sufficient, and 5 = insufficient.

C. Limitations

The statistical strength of the evaluations is low since only 27 students took the final exams in 2008 (16 in 2009) and only ten students filled out the 2009 course's evaluation form. Thus we do not claim to have found reliable evidence that treating MM and FS simultaneously *must* lead to improved understanding of the concepts and better grades in the exams. However, the results do suggest that an increase similar to the one observed in this research is possible.

Readers should also note that the FS contents were not completely identical in both terms (whereas the MM contents were almost identical) since the focus lay on topics which appear in both FS and MM for the 2009 course, whereas no such restriction applied for the 2008 course.

We taught all lectures in the 2008 and 2009 courses except for the FS part in the old course which was taught by a different lecturer. However, in both terms we created exam questions and graded the answers for all topics.

D. Interpretation of the Results

The small increase of +2.32 % in the memory management section is too small to be considered meaningful,

TABLE IV
RESULTS FROM THE STUDENT QUESTIONNAIRE.

	Question / Statement	Avg.	1	2	3	4	5	n/a
1.	Combined treatment of the topics makes sense	1.6	50%	40%	10%	0%	0%	0%
2.	Frequent change between properties of filesystems and memory management is confusing	4.1	0%	10%	10%	40%	40%	0%
	(*Negation of the above statement*)	*1.9*	*40%*	*40%*	*10%*	*10%*	*0%*	*0%*
3.	The combination made it easy to understand that many concepts from one topic translate to the other topic	1.4	60%	40%	0%	0%	0%	0%
4.	More topics should be combined this way	2.8	20%	10%	40%	10%	10%	10%

The meaning of 1–5 is: 1 = agree completely, 2 = agree, 3 = neutral, 4 = disagree, 5 = disagree completely. n/a: not applicable. The questionnaire was filled out by ten students. Statement (2) is also shown negated in line below for better comparison.

especially since the number of test takers was rather small. The increase of +16.3 % in the filesystems section seems significant, but may be influenced by the differences in the FS topic selections and the lecturers who taught the FS parts.

An interesting observation is that in the FS-less course students got slightly higher marks for the MM questions (109.38 %, cf. Table II) than for the other topics, whereas in the two courses held by us they received lower marks for the MM questions (80.56 % / 82.43 %) and average or higher marks for the FS questions (98.28 % / 114.29 %; cf. Table I b). A possible interpretation of this result is that treating closely related topics separately has a worse effect than leaving one of these topics out completely. If this hypothesis could be confirmed, an OS course should either drop the FS topic (and leave it for another term) or combine MM and FS.

VI. CONCLUSIONS AND FURTHER WORK

Results from the double evaluation motivate further research in this area: Since students performed better in the exam and also valued the combined treatment, it makes sense to search for further OS topics that are typically treated separately but might also benefit from being combined. As an example, threads are sometimes only introduced after processes have been fully dealt with, while other lecturers have decided to discuss threads early on during the process introduction. However, processes and threads are not related in the same way that memory management and filesystems are. Instead some knowledge about processes is required to talk about threads, since the latter "live" inside the former.

As to the results presented in this paper, since the classes consisted of only small numbers of students, it would be helpful to repeat this comparison with larger groups of students, ideally with a class large enough that it could be split by pretesting and forming two equally strong groups which then attend lectures that are identical except for the presentation of FS and MM topics.

More generally, it would be interesting to identify further Computer Science topics which are traditionally taught separately but share many concepts, and apply the same approach to their teaching. An example for this (synchronization in OS vs. atomicity of database transactions) was already given above. The overall idea behind this is a shift of focus from concrete topics to general concepts which form the basic building blocks of many problem solutions found in Computer Science.

VII. ACKNOWLEDGEMENTS

We would like to thank Christian Vogt (Munich University of Applied Sciences) for sharing his exam questions and his students' exam results with us and Felix Freiling (University of Erlangen-Nuremberg) for helpfully commenting on a draft version of this paper.

REFERENCES

[1] Christopher, W. A., Procter, S. J., and Anderson, Th. E. 1993. *The Nachos instructional operating system.* In Proceedings of the USENIX Winter 1993 Conference. USENIX Association, Berkeley, CA, USA, 4-4. http://http.cs.berkeley.edu/~tea/nachos/nachos.ps

[2] Claypool, M., Finkel, D., and Wills, C. 2001. *An Open Source Laboratory for Operating Systems Projects*. In Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE) Conference, Canterbury, UK. ftp://ftp.cs.wpi.edu/pub/techreports/pdf/00-24.pdf

[3] Eßer, H.-G. 2009. *Operating Systems*. Munich University of Applied Sciences. Summer Term 2009. Lecture Notes, ch. 6. http://hm.hgesser.de/bs-ss2009/skript/skript-bs-kap06.pdf

[4] Eßer, H.-G. 2009. *Betriebssysteme*. Munich University of Applied Sciences. Summer Term 2009. Slides. http://hm.hgesser.de/bs-ss2009/

[5] Eßer, H.-G. 2011, *Combining Memory Management and Filesystems in an Operating Systems Course* (Poster abstract). In Proceedings of the 16th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2011), June 2011

[6] Freiling, F. C. 2008. *The Design and Implementation of the ULIX Operating System.* Unpublished, available on request.

[7] Hartley, S. J. 1990. *Experience with MINIX in an operating systems lab*, SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education) **22**(3):34–38.

[8] Knuth, D. E. 1999. *MMIXware: a RISC computer for the third millennium*, Springer-Verlag, Berlin, Heidelberg, ISBN: 3-540-66938-8.

[9] Maia, L. P., Machado, F. B., and Pacheco Jr., A. C. 2005. *A constructivist framework for operating systems education: a pedagogic proposal using the SOsim*, ITiCSE '05, pp. 218–222.

[10] van Merriënboer, J. J. G., Clark, R. E., de Croock, M. B. M. 2002. *Blueprints for complex learning: The 4C/ID-Model.* Educational Technology Research and Development, **50**(2):39–64.

[11] Ramakrishnan, S. and Lancaster, A.-M. 1993. *Operating Systems Projects: linking theory, practice and use.* In Proceedings of the 24th SIGCSE technical symposium on Computer science education (SIGCSE '93). ACM, New York, NY, USA, 256-260.

[12] Stallings, W. 2005. *Operating Systems. Internals and Design Principles.* 5th ed. Pearson Education.

[13] Tanenbaum, A. S. 1987. *Operating Systems: Design and Implementation*, Prentice-Hall.

[14] Wagner, T. D. and Ressler, E. K. 1997. *A practical approach to reinforcing concepts in introductory operating systems*. SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education) **29**(1):44–47.