ULIX und Didaktik der Betriebssysteme

Hans-Georg Eßer

Universität Erlangen-Nürnberg
Security Research Group (Informatik 1)

Dagstuhl-Seminar
18.07.–20.07.2011

Hans-Georg Eßer

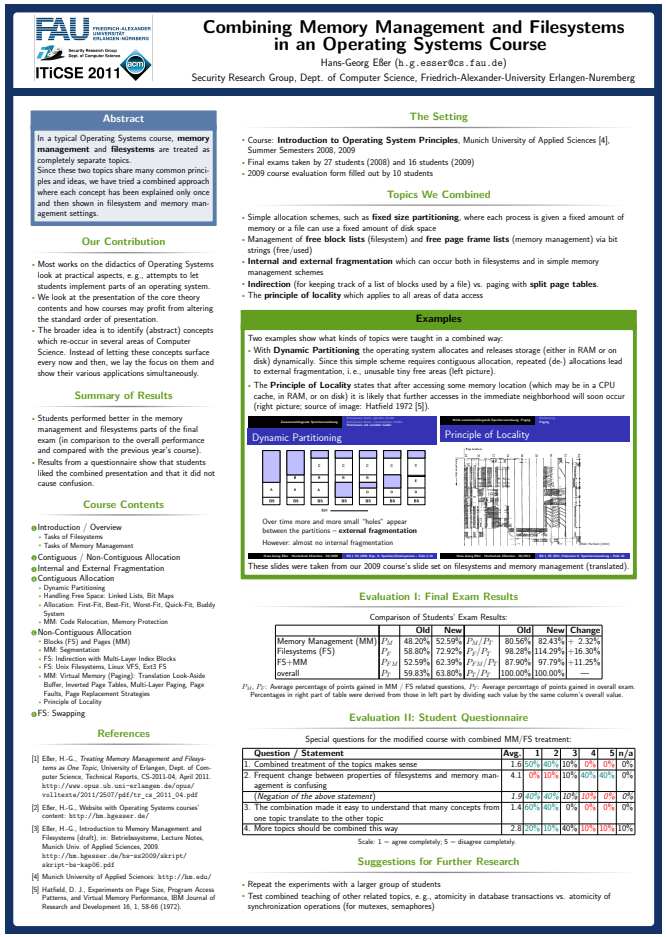
Kurz-CV in <100 Worten

- Studium in Aachen (Mathematik und Informatik)
  - Mathe-Diplom 1997, DA in theor. Informatik
  - Info-Diplom 2005, DA in prakt. Informatik (bei Felix)
- 1999 ein Jahr Doktorand am ZAM, Forschungszentrum Jülich
- seit 2000 Redakteur einer Linux-Zeitschrift (München)
- WS 2006/07, SS 2008, SS 2009, SS 2011: Lehrauftrag „Betriebssysteme“ an der Hochschule München (FH)
- SS 2011, WS 2011/12: Dozent für „Betriebssysteme“ an der FOM Hochschule in München
- 04/2008–09/2010 externer Doktorand, PI1 Mannheim
- seit 10/2010 Mitarbeiter am i1 in Erlangen (50 %)

ULIX, Betriebssysteme, Didaktik

Das letzte Jahr

- Paper zu meiner modifizierten Betriebssysteme- (BS-) Vorlesung
  - als Paper bei ITiCSE 2011 Darmstadt abgelehnt (aber „gut“)
  - als Poster vorgestellt (“Combining Memory Management and Filesystems in an Operating Systems Course”)
  - Paper als Technical Report in Erlangen
- Einarbeitung in ULIX ($\rightarrow$ Felix)
- seit 10/2010 am i1
  - WS 2010/11 und SS 2011: Seminar IT-Sicherheit
  - Betreuung unseres Kursangebots (Univis)



ITiCSE 2011

Combining Memory Management and Filesystems in an Operating Systems Course

Hans-Georg Eßer (h.g.esser@cs.fau.de)
Security Research Group, Dept. of Computer Science, Friedrich-Alexander-University Erlangen-Nuremberg

Abstract

In a typical Operating Systems course, **memory management** and **filesystems** are treated as completely separate topics.
Since these two topics share many common principles and ideas, we have tried a combined approach where each concept has been explained only once and then shown in filesystem and memory management settings.

Our Contribution

- Most works on the didactics of Operating Systems look at practical aspects, e. g., attempts to let students implement parts of an operating system.
- We look at the presentation of the core theory contents and how courses may profit from altering the standard order of presentation.
- The broader idea is to identify (abstract) concepts which re-occur in several areas of Computer Science. Instead of letting these concepts surface every now and then, we lay the focus on them and show their various applications simultaneously.

Summary of Results

- Students performed better in the memory management and filesystems parts of the final exam (in comparison to the overall performance and compared with the previous year's course).
- Results from a questionnaire show that students liked the combined presentation and that it did not cause confusion.

Course Contents

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

References

[1] Eßer, H.-G., *Treating Memory Management and Filesystems as One Topic*, University of Erlangen, Dept. of Computer Science, Technical Reports, CS-2011-04, April 2011. http://www.opus.ub.uni-erlangen.de/opus/volltexte/2011/2507/pdf/tr_ca_2011_04.pdf

[2] Eßer, H.-G., Website with Operating Systems courses' content: http://bm.hgesser.de/

[3] Eßer, H.-G., Introduction to Memory Management and Filesystems (draft), in: Betriebssysteme, Lecture Notes, Munich Univ. of Applied Sciences, 2009. http://bm.hgesser.de/bs-ss2009/skript/skript-ba-kap06.pdf

[4] Munich University of Applied Sciences: http://hm.edu/

[5] Hatfield, D. J., Experiments on Page Size, Program Access Patterns, and Virtual Memory Performance, IBM Journal of Research and Development 16, 1, 58-66 (1972).

The Setting

- Course: **Introduction to Operating System Principles**, Munich University of Applied Sciences [4], Summer Semesters 2008, 2009
- Final exams taken by 27 students (2008) and 16 students (2009)
- 2009 course evaluation form filled out by 10 students

Topics We Combined

- Simple 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
- Management of **free block lists** (filesystem) and **free page frame lists** (memory management) via bit strings (free/used)
- **Internal and external fragmentation** which can occur both in filesystems and in simple memory management schemes
- **Indirection** (for keeping track of a list of blocks used by a file) vs. paging with **split page tables**.
- The **principle of locality** which applies to all areas of data access

Examples

Two examples show what kinds of topics were taught in a combined way:

- With **Dynamic Partitioning** the operating system allocates and releases storage (either in RAM or on disk) dynamically. Since this simple scheme requires contiguous allocation, repeated (de-) allocations lead to external fragmentation, i. e., unusable tiny free areas (left picture).
- The **Principle of Locality** states that after accessing some memory location (which may be in a CPU cache, in RAM, or on disk) it is likely that further accesses in the immediate neighborhood will soon occur (right picture; source of image: Hatfield 1972 [5]).

Dynamic Partitioning

Principle of Locality

Over time more and more small "holes" appear between the partitions – **external fragmentation**

However: almost no internal fragmentation

These slides were taken from our 2009 course's slide set on filesystems and memory management (translated).

Evaluation I: Final Exam Results

Comparison of Students' Exam Results:

		Old	New		Old	New	Change
Memory Management (MM)	P_M	48.20%	52.59%	P_M / P_T	80.56%	82.43%	+ 2.32%
Filesystems (FS)	P_F	58.80%	72.92%	P_F / P_T	98.28%	114.29%	+16.30%
FS+MM	P_{F+M}	52.59%	62.39%	P_{F+M} / P_T	87.90%	97.79%	+11.25%
overall	P_T	59.83%	63.80%	P_T / P_T	100.00%	100.00%	—

P_M, P_F: Average percentage of points gained in MM / FS related questions, P_T: Average percentage of points gained in overall exam. Percentages in right part of table were derived from those in left part by dividing each value by the same column's overall value.

Evaluation II: Student Questionnaire

Special questions for the modified course with combined MM/FS treatment:

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%

Scale: 1 = agree completely; 5 = disagree completely.

Suggestions for Further Research

- Repeat the experiments with a larger group of students
- Test combined teaching of other related topics, e. g., atomicity in database transactions vs. atomicity of synchronization operations (for mutexes, semaphores)

ULIX – Stand der Dinge

ULIX

- ULIX = virtuelle Hardware + darauf laufendes BS
- „Minix-Buch, aber mit Literate Programming“
- Felix: Beschreibung der HW, Grundfunktionen des BS
- Bachelor- und Seminararbeiten:
  - Emulator für die ULIX-Hardware (Ralf Hund, Romi Sorge)
  - Assembler für die ULIX-Maschinensprache (Nadine Benedum)
  - Compiler als GCC-Portierung (Balthasar Biedermann)
  - Übersicht Lehr-BS (Tobias Kienzle)

Literate Programming

Andere Herangehensweise an Dokumentation

- *nicht*: Code + Dokumentation (à la JavaDoc)
- *sondern*: Dokumentation (Buch), mit integriertem Code

To service an interrupt, we need to "simulate" a TRAP command (see Section 2.5.3): We save the MODE bit and the return address to the system stack, switch to system mode and then jump to the interrupt handler.

27a ⟨*service interrupt within instruction cycle* 27a⟩≡

```
MM[SSP--] = MODE;
MM[SSP--] = PC;
MODE = SYSTEM_MODE;
PC = interrupt_table[IRR];
```

We now see how TRAP k can be implemented. The parameter k is the type of interrupt (or the level of the interrupt) which should be raised. So if we want to do a TRAP it suffices to set IRR to the value k.

27b ⟨*execute* TRAP k 27b⟩≡

```
IRR = k;
```

- Konzept von Knuth ("TEX: the program")

ULIX – Probleme

Bisherige Entwicklungsumgebung

- ULIX-Tool-Chain problematisch:
  - Compiler und Assembler nicht mit echtem Code getestet
  - ULIX-BS-Code kompiliert (bis auf Inline-Assembler), aber bisher ohne Funktion
  - Assembler/Compiler erzeugen kein link-bares Object-Format, sondern Speicher-Image für Tests im Emulator
- $\rightarrow$ wo ansetzen?

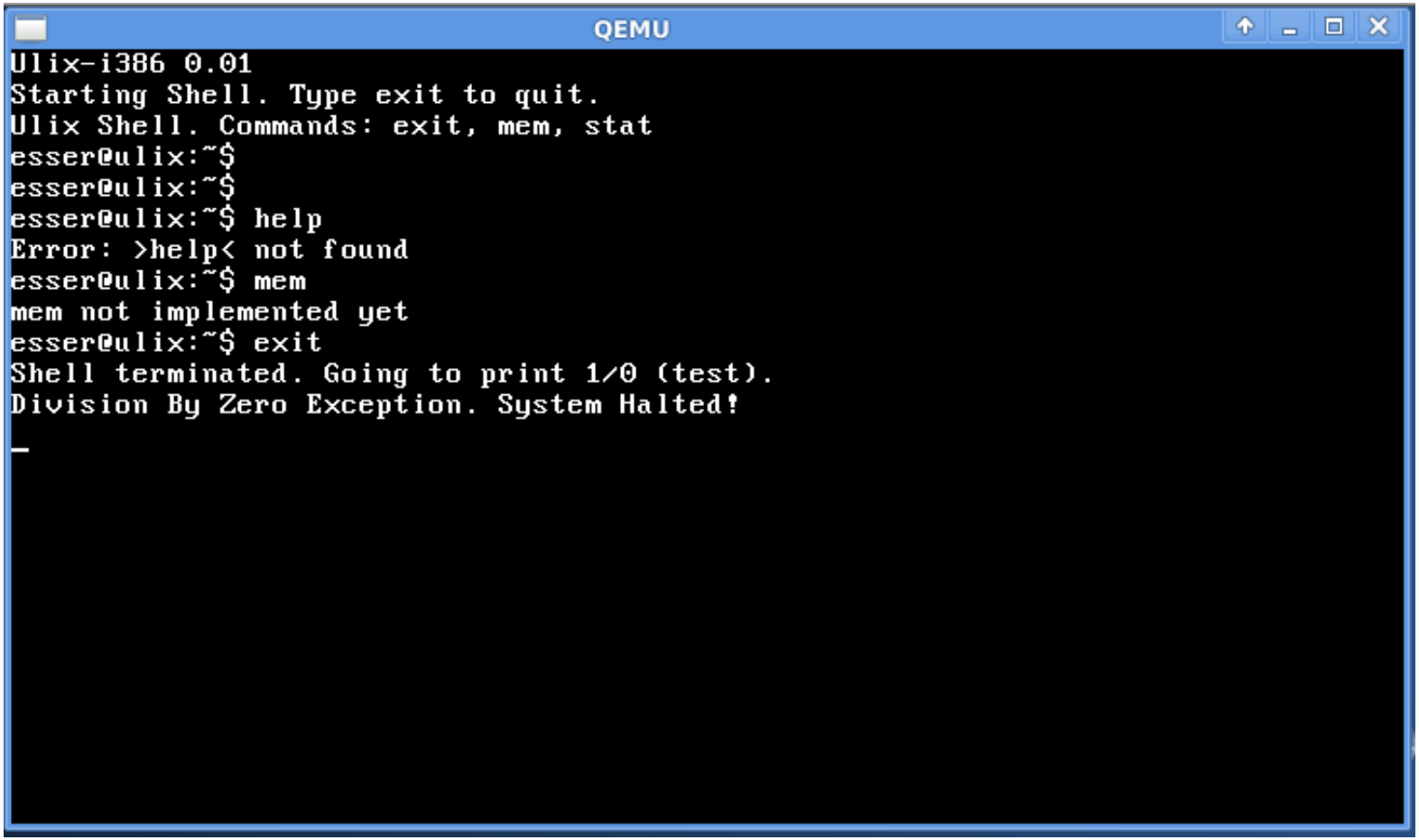
ULIX – mein aktueller Ansatz: Intel

ULIX-Portierung auf Intel-Architektur

- Basis: BS-Workshops im Netz
  - `http://www.jamesmolloy.co.uk/tutorial_html/`
  - `http://www.osdever.net/tutorials/view/brans-kernel-development-tutorial`
  - Informationen auf `http://osdev.org`
- Intel-Welt bringt eigene (Legacy-) Probleme mit:
  - Segmentierung – will man nicht, muss man aber
  - Interrupts: irre Legacy-Implementierung mit Master- und Slave-Interrupt-Controller und obskurem Mix aus HW- und SW-Interrupts
- aber: große Intel-OS-Developer-Community :)

ULIX-Intel

- Protected Mode, Interrupts, Mini-Shell



ULIX-Intel – Entwicklungsplattform

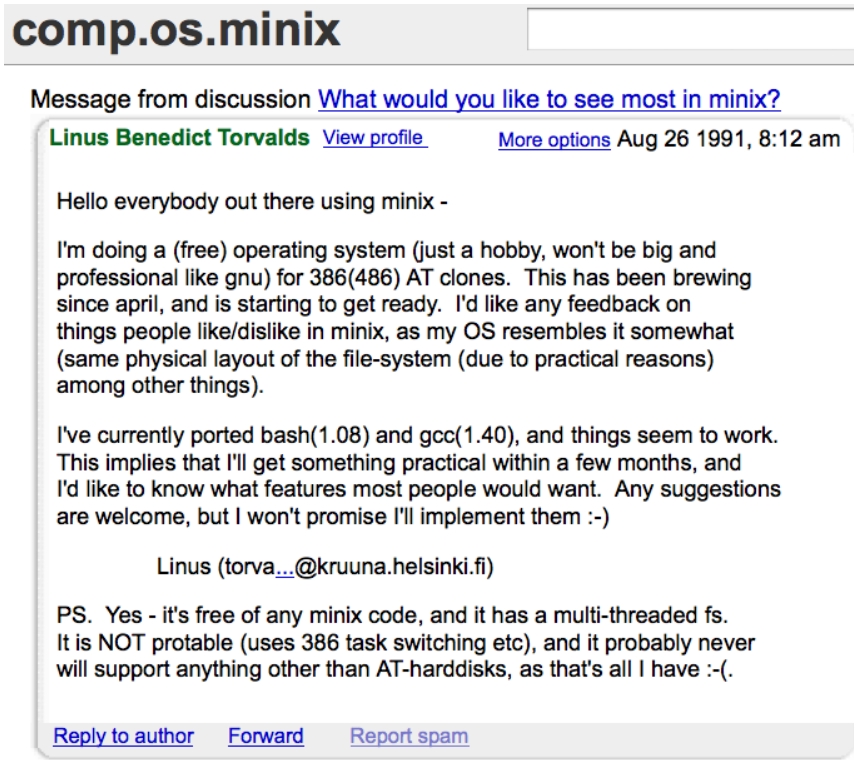
Standard-Tool-Chain

- Linux
- `gcc`, `nasm`
  (`nasm` hat besser lesbare Syntax als GNU-Assembler)
- Tests in `qemu`
  $\rightarrow$ kann Kernel starten, ohne Boot-Disk zu erzeugen

ULIX – Pläne

So soll es weiter gehen:

- Bis Ende des Jahres: Lauffähiges ULIX-Intel mit ...
  - Prozess- und Speicherverwaltung
  - Round Robin Scheduler
  - einfaches Dateisystem
  - Shell
  - dabei so viel Code wie möglich aus „ULIX-Ulix“ übernehmen
- später: Portierung auf ULIX-Hardware



comp.os.minix

Message from discussion What would you like to see most in minix?

Linus Benedict Torvalds View profile More options Aug 26 1991, 8:12 am

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torva...@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Reply to author Forward Report spam

ULIX – didaktischer Hintergrund

Fragestellung: Literate Programming und BS

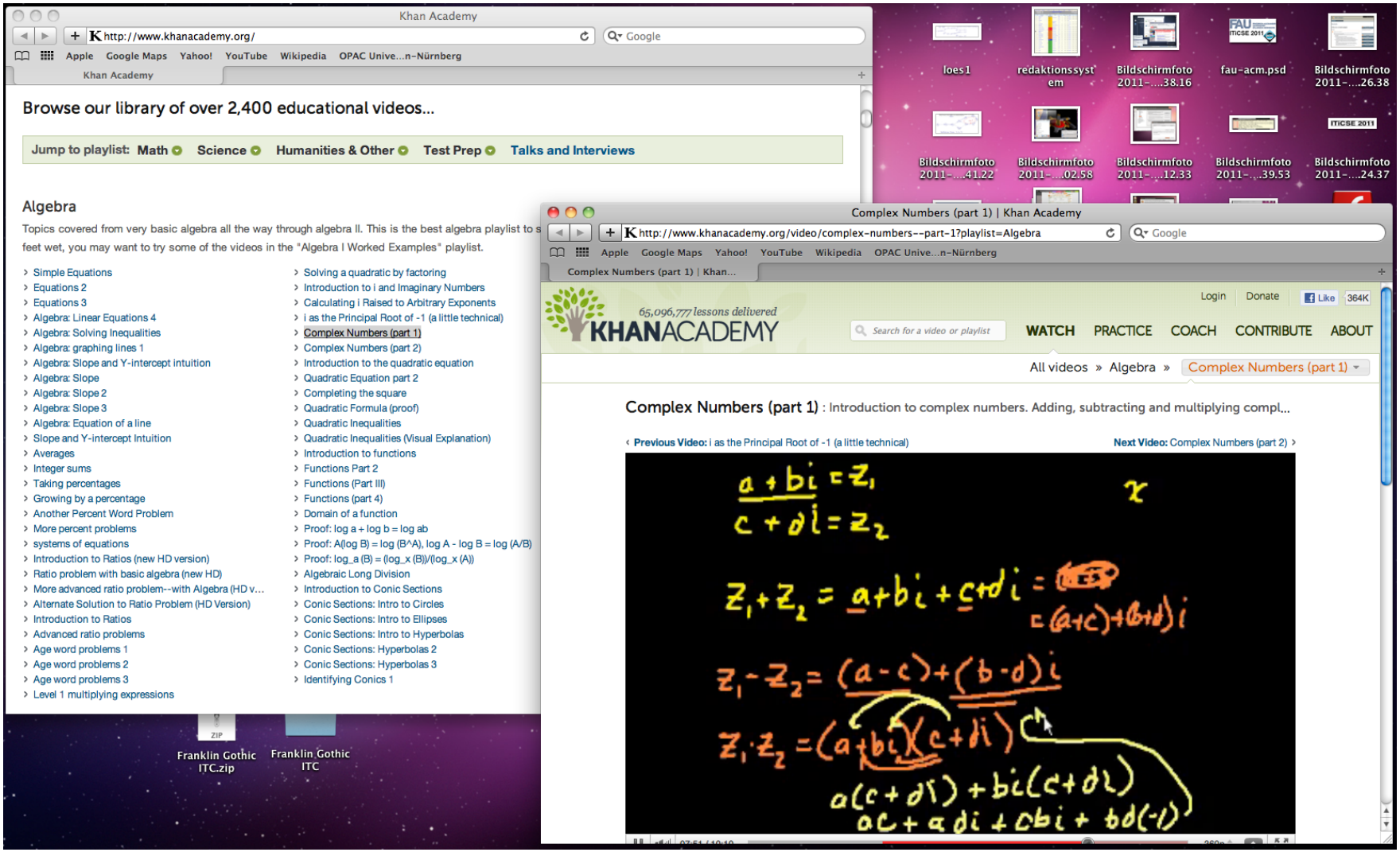
- Wie gut eignet sich ein mit Literate Programming erstelltes BS für die Vermittlung der BS-Theorie?
- nach Fertigstellung der Implementierung „Test“ im Rahmen einer BS-Vorlesung
- Evaluierung der Vorlesung
- Diss. enthält technische Beschreibung der ULIX-Entwicklung und Ergebnisse des Praxistests in der Vorlesung

Alternative zu Vorlesung

Videos

- Produktion einer Serie von 10-Minuten-Videoschnipseln
- vollständige Darstellung der BS-Theorie anhand von ULIX
- Videos auf YouTube mit begleitender Webseite
  $\rightarrow$ Programmieraufgaben, Literatur etc.
- Vorbild: "Khan Academy" (`http://khanacademy.org/`)

Khan Academy



Konferenzen?

Wenig Angebot ...

- Didaktik:
  - SIGCSE (ACM Technical Symposium on Computer Science Education), USA
  - ITiCSE (ACM Conf. on Innovation and Technology in Computer Science Education), Europa
  - Forschung überwiegend im Bereich Didaktik der Informatik an Schulen (nicht Hochschulen)
- vielleicht auch Betriebssysteme:
  - OSDI (USENIX Symposium on Operating Systems Design and Implementation), 2-jährlich → Okt. 2012