Organization

Advanced Operating Systems

Meet the AOS Team

Lecturers

Cristiano Giuffrida *Erik van der Kouwe*

Teaching Assistants

Sebastian Österlund *Manuel Wiesinger*

Contacts

<aos@vusec.net>

https://vusec.net

Join Us!

Teaching Assistantships Available

https://www.vusec.net/join/#ta

Capture The Flag

<vubar@vusec.net>

The “When” and “Where”

Lectures:

- Tuesday, Friday (extra Tue, Thu first week)
- Check Google Calendar on Canvas for time slots

Exam:

- None

Policies:

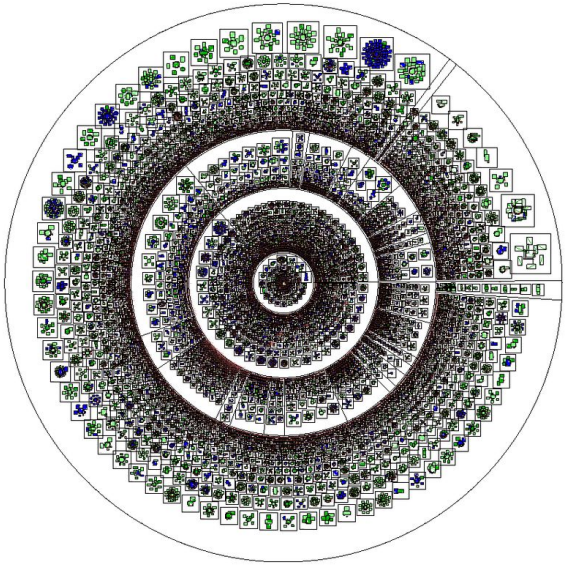
- All the lectures are “online” this year (details later)
- Attendance not mandatory, but strongly encouraged
- Attendance mandatory for grading sessions

The “What”

BYOOS:

Build Your Own Operating System

(From the ground up)



The “Why”

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. [...]

Linus (torvalds@kruuna.helsinki.fi)

The “Why”

- The OS monoculture days are numbered
  - 2009: **seL4** [*SOSP*]: formally verified OS kernel
  - 2012: **OKL4** on 1.5 billion devices
  - 2015: MS **IoT Core**, Google **Brillo**, Huawei **LiteOS**
  - 2018: **Biscuit** [OSDI], **Tock** OS [SOSP]
- You can build crazy little things!
  - 2010: **Loris** [DSN]: dependable storage stack
  - 2012: **NetwOS** [*DSN*]: fast & reliable network stack
  - 2013: **ProteOS** [*ASPLOS*]: whole-OS live update
  - 2016: **OSIRIS** [*DSN*]: whole-OS crash recovery
  - 2017: **VUsion** [SOSP]: Safe memory deduplication
  - 2018: **ZebRAM** [OSDI]: Full Rowhammer protection
  - 20xx: More… (www.vusec.net)

The “Why”

- Learn how (low-level) “stuff works”
- Look at real-world systems code
- Learn the kernel programming workflow
- Get insights into modern OS research
- Brew your own kernel

The “How”

- Two common “Advanced OS” models:
  - Learn & hack **Linux kernel** code
    - Real code, but only toy modifications possible
  - Hack a **Toy** operating system
    - Arbitrary modifications, but only learn toy code

The “How”

- Two common “Advanced OS” models:
  - Learn & hack **Linux kernel** code
    - Real code, but only toy modifications possible
  - Hack a **Toy** operating system
    - Arbitrary modifications, but only learn toy code
- Enter Advanced OS at VU:
  - Learn **Linux kernel**, code on **your own OS**!
    - https://elixir.bootlin.com/linux/v5.8/source

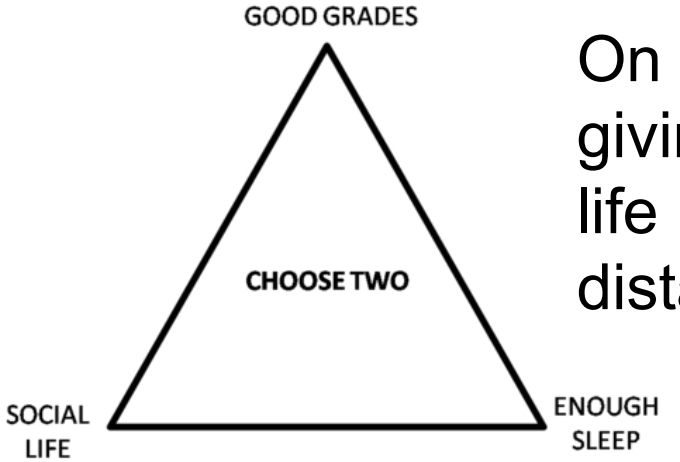
The “How”

- Course based on the JOS framework
- Boilerplate code to build your own kernel
- JOS originally developed at MIT
- We’ll use a modified version at VU
- AKA OpenLSD
  - Open Learning Software Distribution

The “How”

WARNING!

- This course is **hard** (really)...
- Only take it if you enjoy **hard-to-find bugs** :)
- The **MIT model**:



On the bright side: giving up your social life is great for social distancing

The “How”

What they say about AOS:

“Wait, this looks like too much work!”

Andrea Arcangeli

- Core Linux kernel MM developer

The “How”

But...

- This course is also very **rewarding**
- No other course will let you:
  - Learn everything about kernel hacking
  - Solve crazy little challenges
  - Collect customized bonuses down the line
  - Build your own LSD!

The “How”

What YOU say about AOS:

“My favourite course at the VU”

“A super fun and challenging course!”

“The best course in my entire study”

“I learned a ton, super solid job guys!”

The “How”

WARNING!

- This course is very **“experimental”**
- And **you** are part of the experiment!
- New features & assignments every year

- Planning to also do CNS? Warnings++

Course Material

Canvas [https://canvas.vu.nl/courses/49330]

- Registration
- Slides
- Recorded video lectures
- Course info
- Announcements

Textbook

- `kernel.c`

Course Structure

- **Hands-on implementation of an OS kernel**
  - Labs based on OpenLSD
  - Basic OS subsystems
  - Need to know C, basic OS/architecture principles

- **Lectures on OS subsystems**
  - Necessary for the assignments (**labs**)
  - Sample code from Linux kernel and OpenLSD
  - Discussion on other research OSes

Labs: Structure

- Core lab + discussion (9 points)
- Bonus(es) (e.g., +2 points) for each lab
- Grading based on lab **and** bonus(es)
- You can get up to 11.0 per lab
  - Which we'll average and cap the final grade to 10.0

Lectures: Main topics

1. **Memory management**
   a. Physical memory
   b. Virtual memory
2. **Process management**
   a. Address spaces (user/kernel)
   b. Interrupts, system calls
3. **Multiprocessing**
4. **Multicore**
5. **Other advanced topics**

Labs: Structure

- Every Friday:
  - Current hard Lab deadline, 23:59 (-1 each day late)
  - More time for Lab 7 (see Calendar)
  - Canvas submission (more details later)
- But also:
  - Grading current Lab
  - Intro to next Lab
- This means:
  - **Lab 1 deadline on Friday, 23:59**

Labs: Teams

- You are encouraged to work in teams
- Kernel programming is a team activity
  - Linux kernel: ~30 MLOC → ~15,000 developers
  - OpenLSD kernel: ~5 KLOC → 2 kernel developers
- Find your team mate
- Each team of 2 registers *team* on Canvas
  - No later than Lab 1 deadline (ideally asap)
  - Note: register team even if you want to work alone
  - More on this in Intro to Lab 1

Labs:
How to Work in Teams

Lab work:

- Try to pick a teammate with **similar** skill set
- Split work **equally**, as much as possible
- Need to know **entire** code of Lab submitted
- Separate the bonus code from the rest of the Lab code using, for instance, for Lab 1:

```
#ifdef BONUS_LAB1
#endif
```

Labs:
How to Work in Teams

Discussion:

- N questions directed to each team member
- Be prepared to also answer questions about your teammate's code
- Questions will also cover the design of bonus assignments
- Even those you did not implement

Grading

Labs (100%):

- Every Friday

Requirements to pass the course:

- At least 4.0 for each Lab
- At least 5.5 overall (after averaging)
- -1.0 per day late

No resit for this course!

Course Schedule (First Week)

Tuesday (11:00-on):

- Course organization and setting up
- Frame allocation
- Intro to Lab 1 on frame allocation

Wednesday (11:00):

- GDB walkthrough, help on Lab 1

Friday (13:30):

- Page tables
- Discussion Lab 1, Intro to Lab 2

Friday (23:59):

- Deadline Lab 1

Course Schedule (Starting Second Week)

Tuesday (11:00):

- Interrupts and process management

Friday (13:30):

- Discussion Lab 2
- Intro to Lab 3

Friday (23:59):

- Deadline Lab 2

Tentative Lab Deadlines

1. *You got framed (*Frame allocation*)*
   Deadline: 04.09.2020
2. *On the same page (*Page tables*)*
   Deadline: 11.09.2020
3. *Three rings under (*User-mode support*)*
   Deadline: 18.09.2020
4. *Don't stop me now (*Paging*)*
   Deadline: 25.09.2020
5. *All together now (Multiprocess support)*
   Deadline: 02.10.2020
6. *The more, the merrier (Multicore support)*
   Deadline: 09.10.2020
7. *Under Pressure (Out-of-memory management)*
   Deadline: 16.10.2020 (soft), 23.10.2020 (hard)

On Plagiarism

WARNING!

- **Plagiarism**: we take this seriously!

- What is **plagiarism** in AOS (+other courses)?
  - Copy (part of) a solution from **another team**
  - Copy (part of) a solution from the **Internet**
  - Buy a solution from **any other source**
  - **Copy + make minor changes** to any of the above

On Plagiarism

WARNING!

- **Plagiarism**: we take this seriously!

- Why is **plagiarism** (really) bad?
  - Bad for **you**: hurts your learning process
  - Bad for **others**: takes away from the work of others
  - Bad for **us**: a lot of extra unthankful work
  - Bad for **everybody**: hurts reputation of course & VU

On Plagiarism

WARNING!

- **Plagiarism**: we take this seriously!

- How is **plagiarism** detected in AOS?
  - **Automatic** plagiarism detection tools
    - Other current / old teams
    - Internet
  - **Manual** checks

On Plagiarism

WARNING!

- **Plagiarism**: we take this seriously!

- What happens if you **commit plagiarism**?
  - You will be **reported** to the exam committee
  - It is up to them to decide on **disciplinary actions**
  - **Good news**: you get exactly 1 warning first
  - **Bad news**: that warning is today!

AOS in Corona Times

First things first:

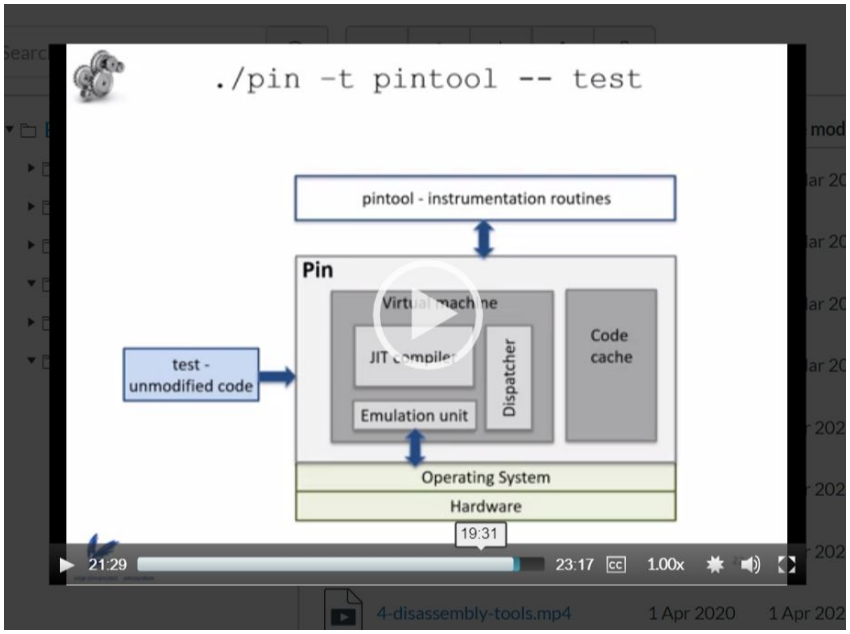
- If you are affected by Corona in any way, do seek help
- The student advisor is Natalia <n.silvis-cividjian@vu.nl>
- Stay safe!

Everything changes, but nothing really changes:

- No f2f lectures, but we'll move *everything* online
- Assignment discussion is also online
- Calendar with **tentative** schedule on Canvas
- Based on the *blending learning model*

AOS in Corona Times

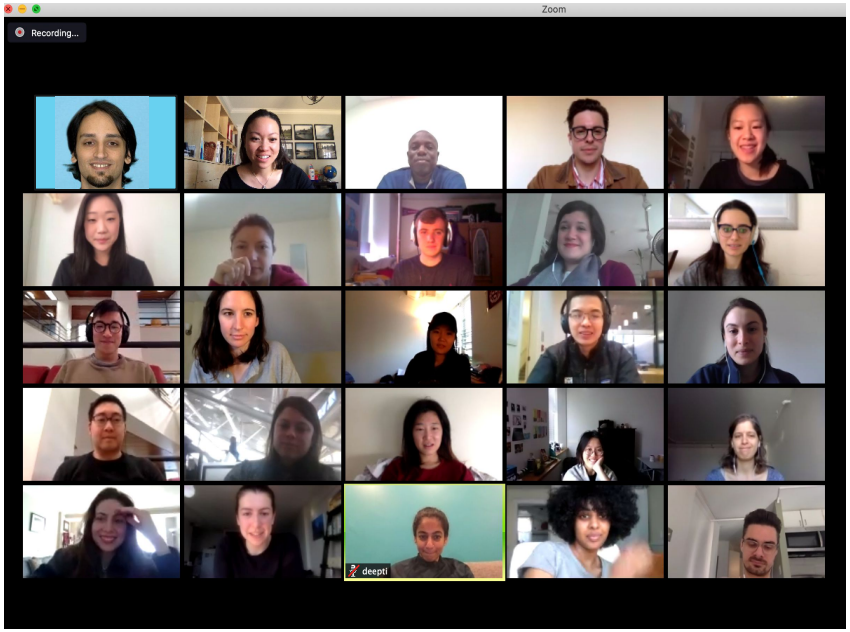
- **Blended learning model in short:**



Recorded lecture online 1 day before

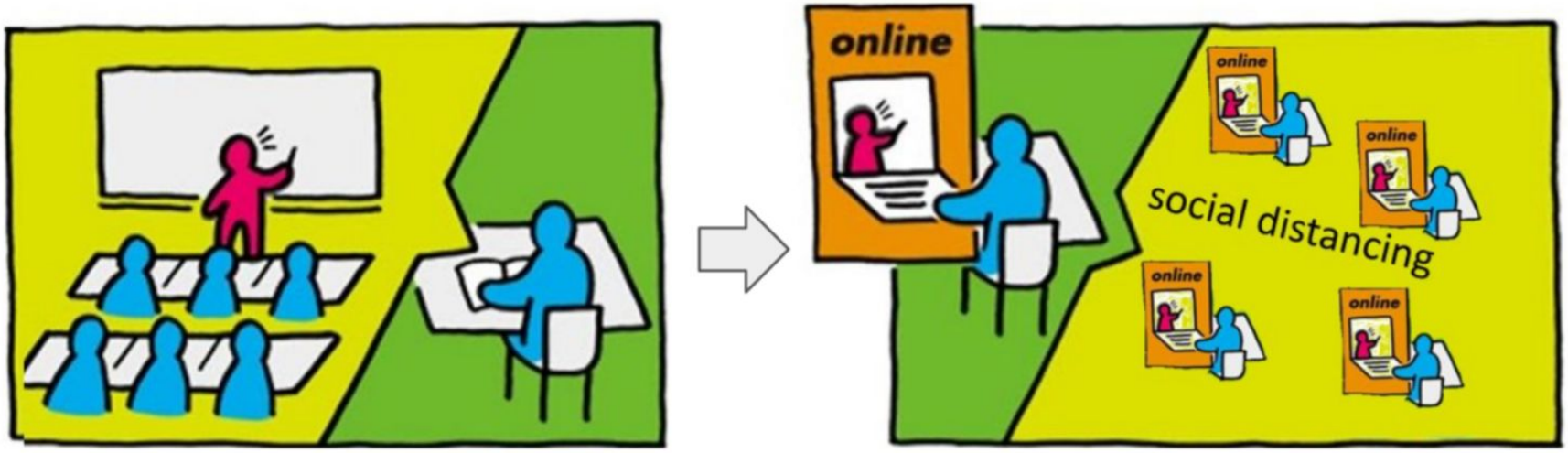
AOS in Corona Times

- **Blended learning model in short:**



Virtual classroom in the lecture slot

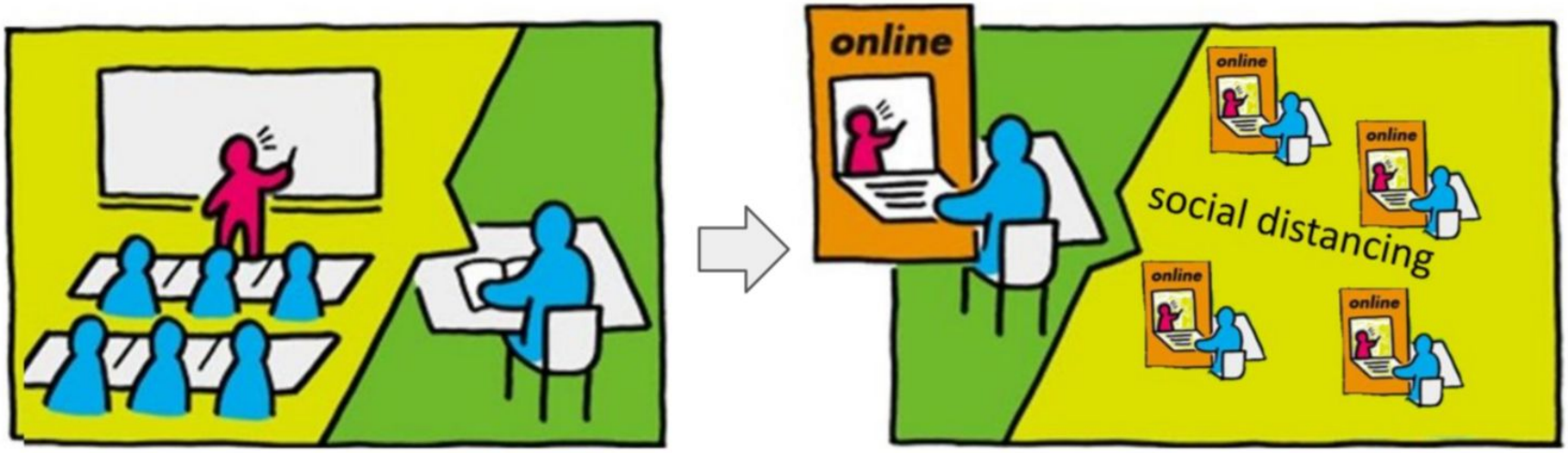
Blending Learning Model



Regular lecture presentations available on Canvas

- Recorded presentation with voice-over (Files -> *videos*)
- Slides (Files -> *slides*)
- At least 1 day before the corresponding lecture slot
- Make sure to study the presentation in advance

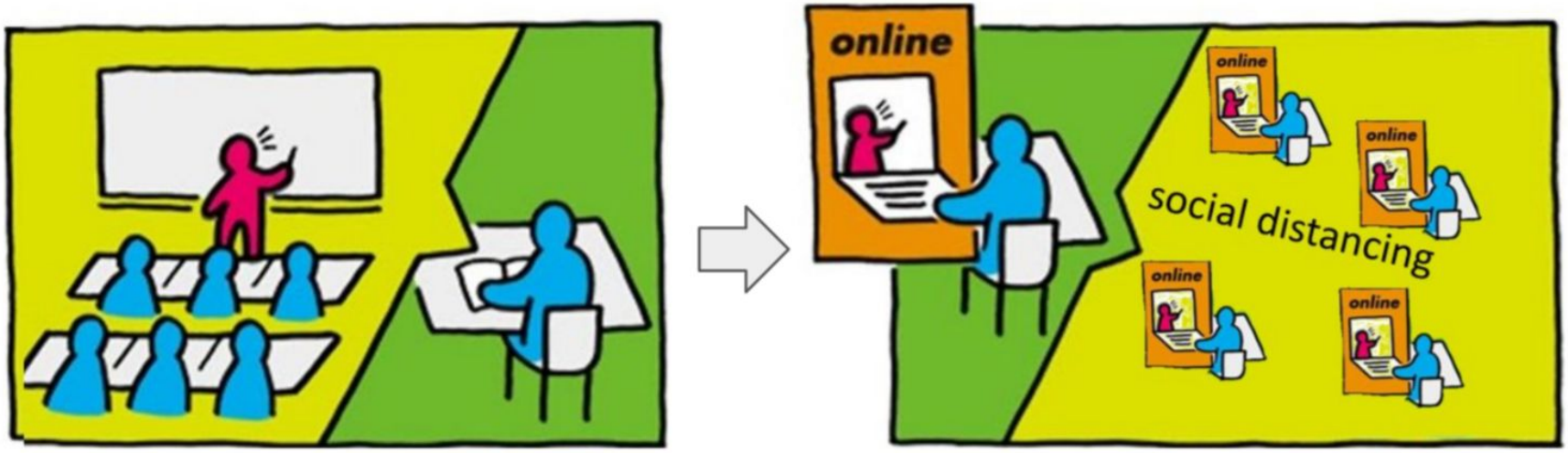
Blending Learning Model



Live Q&A during the regular lecture slot

- For the regular lecture material
- Aim for ~30 minutes (unless there is reason to extend)
- *Zoom* link for live video (web-based) - via VU account
  - See Calendar on Canvas for link

Blending Learning Model



Labs

- Introduced live using the same *Zoom* link
- We'll switch to *Slack* for the discussion sessions on Friday
  - See Canvas to sign up to the vu-aos *Slack*

Teaching Assistants

Sebastian Österlund

Manuel Wiesinger

Help with the Labs

For public questions safe to share with other students:

- **Always** use the Canvas discussion board

For private questions (e.g., about your own solution):

- Email us at aos@vusec.net - we all read this
- TAs will host "office hours" on vu-aos *Slack* for live help
- For now, we will start "office hours" at the end of live Q&As, so make sure you attend them