Operating Systems

Daniel Gruss

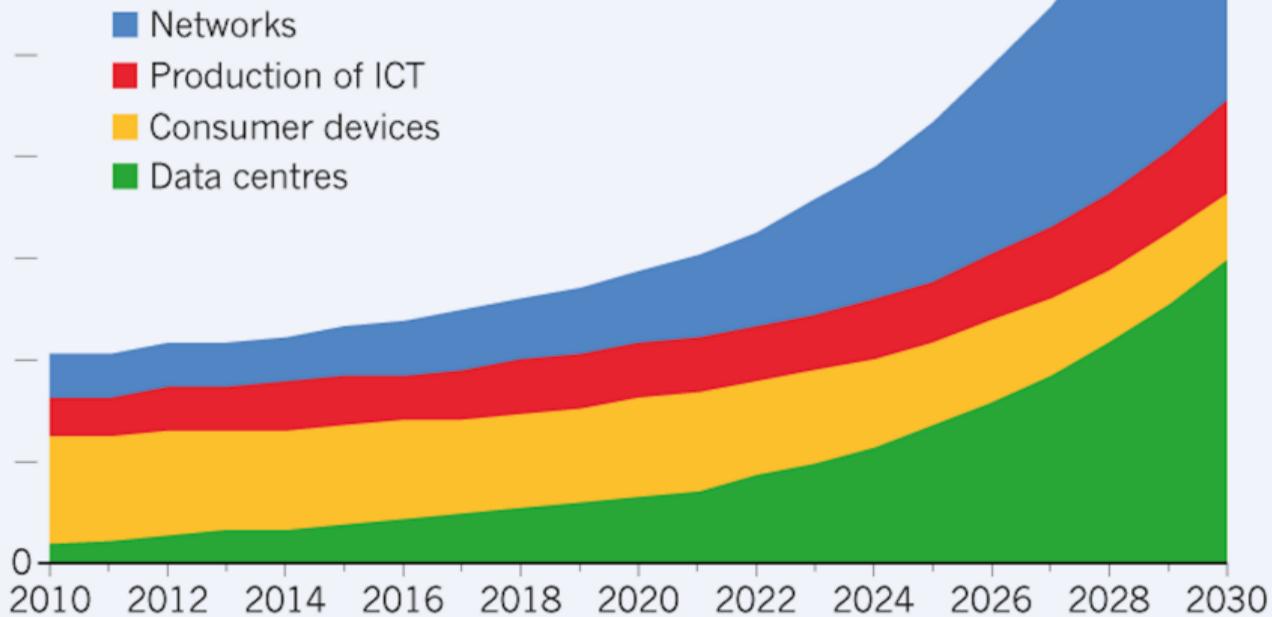2024-03-01

Why OS?

ENERGY FORECAST

9,000 terawatt hours (TWh)

©nature

20.9% of projected electricity demand

- Networks
- Production of ICT
- Consumer devices
- Data centres

2010 2012 2014 2016 2018 2020 2022 2024 2026 2028 2030

It's 2030. You get to make an OS design decision:
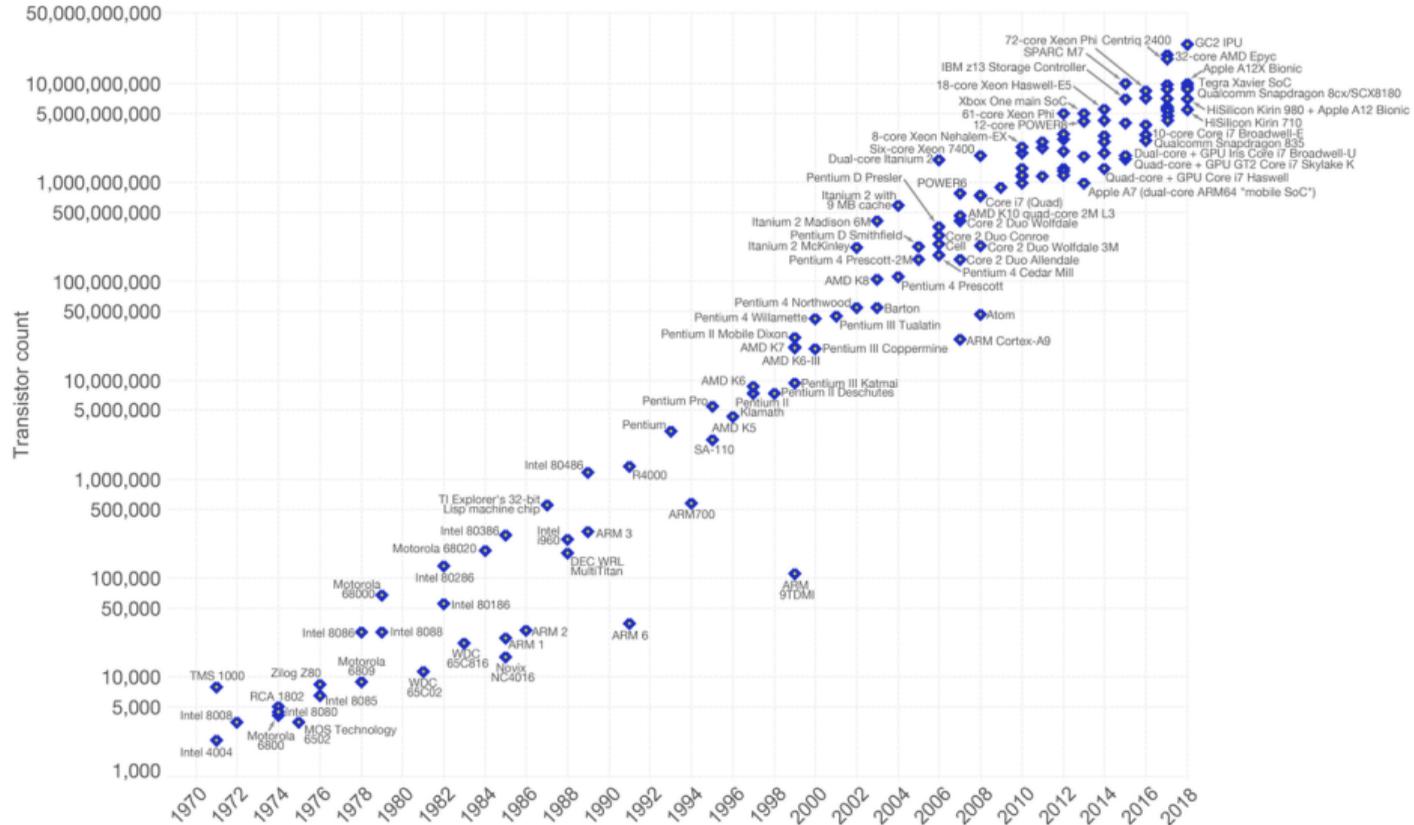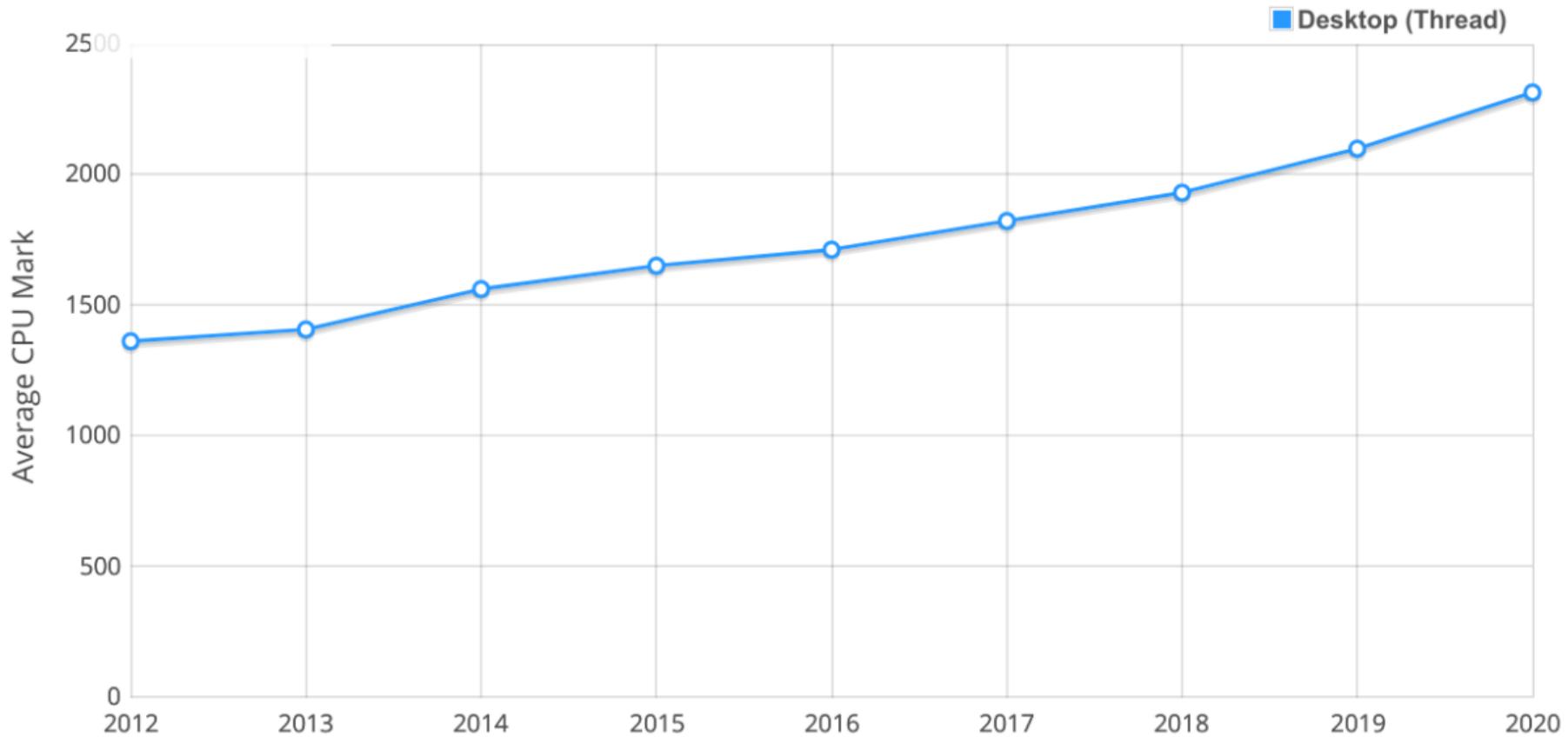
one page table per process

vs

half a page table per process + 1 for the kernel

0.1%

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are
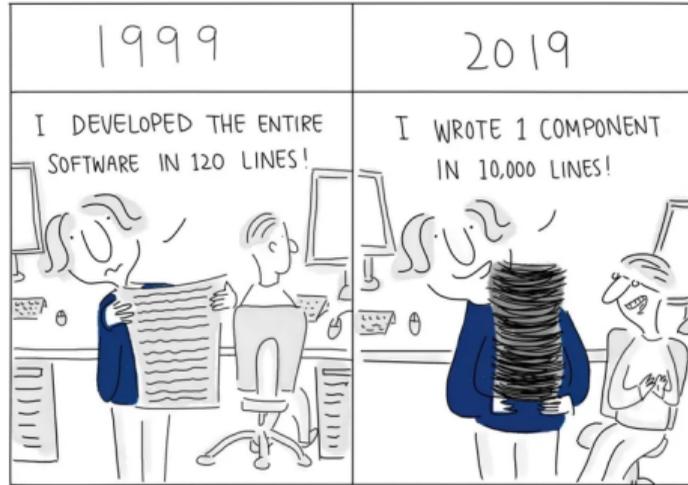linked to Moore's law.

Transistor count

50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000

72-core Xeon Phi Centriq 2400
SPARC M7
IBM z13 Storage Controller
18-core Xeon Haswell-E5
Xbox One main SoC
61-core Xeon Phi
12-core POWER8
8-core Xeon Nehalem-EX
Six-core Xeon 7400
Dual-core Itanium 2
Pentium D Presler
POWER6
Itanium 2 with
9 MB cache
Itanium 2 Madison 6M
Pentium D Smithfield
Itanium 2 McKinley
Pentium 4 Prescott-2M
AMD K8
Pentium 4 Northwood
Pentium 4 Willamette
Pentium II Mobile Dixon
AMD K7
Pentium II Mobile Dixon
AMD K6-III
AMD K6
Pentium III Katmai
Pentium Pro
Pentium II Deschutes
Klamath
AMD K5
Pentium
SA-110
Intel 80486
R4000
TI Explorer's 32-bit
Lisp machine chip
Intel 80386
Motorola 68020
Intel
80286
DEC WRL
MultiTitan
Motorola
68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
ARM 1
ARM 3
Motorola
68000
Intel 80286
Motorola
6809
RCA 1802
Intel 8085
Intel 8080
MOS Technology
6502
TMS 1000
Zilog Z80
Intel 8008
Motorola
6800
Intel 4004
WDC
65C816
Novix
NC4016
WDC
65C02
ARM 6
ARM
9TDMI

GC2 IPU
32-core AMD Epyc
Apple A12X Bionic
Tegra Xavier SoC
Qualcomm Snapdragon 8cx/SCX8180
HiSilicon Kirin 980 • Apple A12 Bionic
HiSilicon Kirin 710
10-core Core i7 Broadwell-E
Qualcomm Snapdragon 835
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Apple A7 (dual-core ARM64 "mobile SoC")
Quad-core + GPU Core i7 Haswell
Core i7 (Quad)
AMD K10 quad-core 2M L3
Core 2 Duo Wolfdale
Core 2 Duo Conroe
Cell
Core 2 Duo Wolfdale 3M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
Pentium 4 Prescott
Atom
Barton
Pentium III Tualatin
ARM Cortex-A9
Pentium III Coppermine

1970 1972 1974 1976 1978 1980 1982 1984 1986 1988 1990 1992 1994 1996 1998 2000 2002 2004 2006 2008 2010 2012 2014 2016 2018

SWEB is C and C++

Which programming language would you prefer?

Table 4. Normalized global results for Energy, Time, and Memory

	Total					

	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(c) Lisp	2.27	(c) Pascal	3.02	(c) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(c) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84
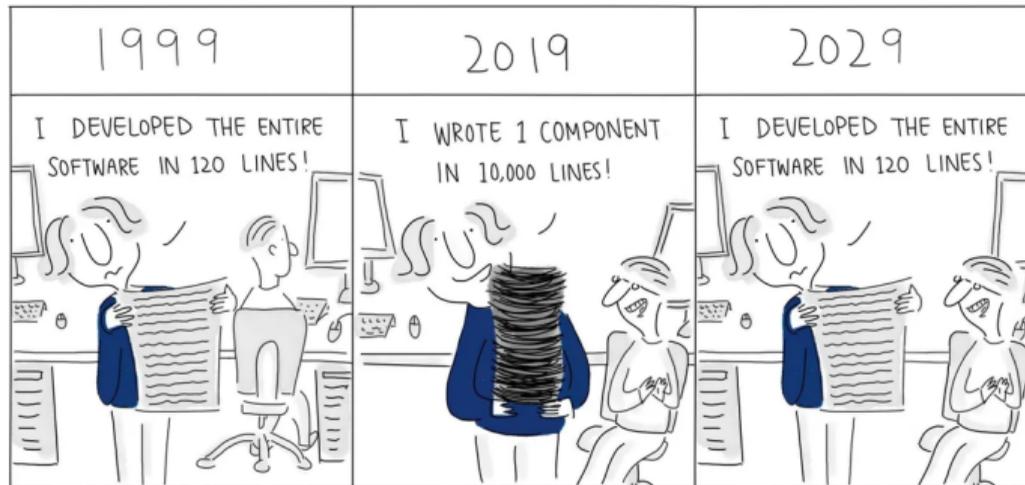
You and SWEB

Organizational Details

SWEB

Booting SWEB

Back to Assignment 1

You and SWEB

Daniel Gruss

- Knowledge from earlier lectures
  - ESP, OOP1(!), SLP(!!), CON, ...
  - **Everything** taught in SLP, OOP1, and ESP will be practically applied **by you** in OS

- Knowledge from earlier lectures
  - ESP, OOP1(!), SLP(!!), CON, …
  - **Everything** taught in SLP, OOP1, and ESP will be practically applied **by you** in OS
- Reasonable C/C++ experience
- Team work + Time management
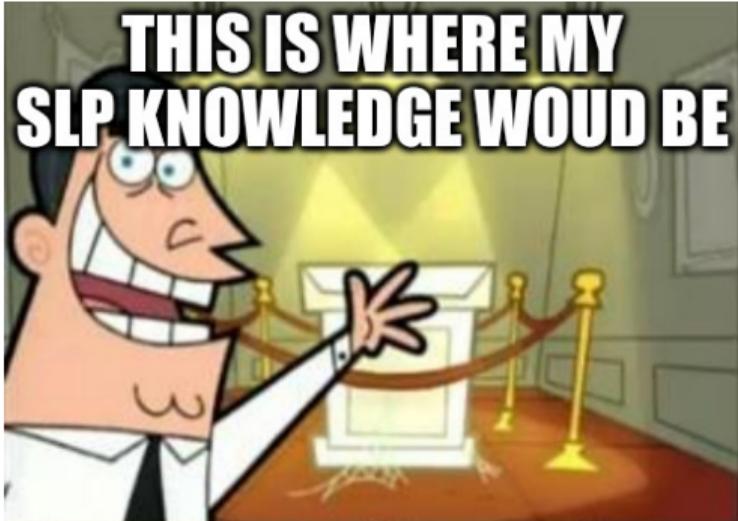
Daniel Gruss

THIS IS WHERE MY SLP KNOWLEDGE WOUD BE

IF I HAD ONE

- Depends significantly on your knowledge and experience ...

Daniel Gruss

- Depends significantly on your knowledge and experience …

- … and on your team members.

Daniel Gruss

- Depends significantly on your knowledge and experience …

- … and on your team members.

- Short good solutions add around 4000 lines of code $\rightarrow$ 1000 lines of code per member

- Depends significantly on your knowledge and experience …
- … and on your team members.
- Short good solutions add around 4000 lines of code $\rightarrow$ 1000 lines of code per member

Daniel Gruss

- No one should be afraid of OS.

Daniel Gruss

- No one should be afraid of OS.
- We even added a course to prepare you for OS (and other courses): SLP!

Daniel Gruss

- No one should be afraid of OS.
- We even added a course to prepare you for OS (and other courses): SLP!
- $\rightarrow$ SLP is sort of the "'booster"' for OS

- No one should be afraid of OS.

- We even added a course to prepare you for OS (and other courses): SLP!

$\rightarrow$ SLP is sort of the "'booster"' for OS

- Statistics confirm: if you did well in SLP, OS becomes very much doable

Daniel Gruss

"I'm afraid!"

- No one should be afraid of OS.
- We even added a course to prepare you for OS (and other courses): SLP!
- → SLP is sort of the "'booster"' for OS
- Statistics confirm: if you did well in SLP, OS becomes very much doable
- Steep learning curve in the beginning?

Daniel Gruss

"I'm afraid!"

- No one should be afraid of OS.
- We even added a course to prepare you for OS (and other courses): SLP!
- → SLP is sort of the "'booster"' for OS
- Statistics confirm: if you did well in SLP, OS becomes very much doable
- Steep learning curve in the beginning?

Daniel Gruss

"I'm afraid!"

- No one should be afraid of OS.
- We even added a course to prepare you for OS (and other courses): SLP!
- → SLP is sort of the "'booster"' for OS
- Statistics confirm: if you did well in SLP, OS becomes very much doable
- Steep learning curve in the beginning? No worries, you will manage!

Daniel Gruss

Note to students

While we have made an effort to simplify these projects for you, most Duke students find these projects sufficiently difficult to dominate their lives during the one-semester course.

Note to students

While we have made an effort to simplify these projects for you, most Duke students find these projects sufficiently difficult to dominate their lives during the one-semester course. A common misconception from earlier semesters is that we are sadistic individuals who enjoy seeing students suffer.

Daniel Gruss

Note to students

While we have made an effort to simplify these projects for you, most Duke students find these projects sufficiently difficult to dominate their lives during the one-semester course. A common misconception from earlier semesters is that we are sadistic individuals who enjoy seeing students suffer. Actually, this is not the case. We enjoy seeing students who are proud of what they have accomplished and excited by the power that flows from a relatively small set of simple abstractions in an operating system, even a toy one like Nachos.

Wie beurteilen Sie Ihren für diese LV nötigen Arbeitsaufwand - gemessen an den ECTS-Punkten, die Sie erhalten? /
How would you rate the workload for this course - measured against the ECTS credit points you received? (71 x answered)

sehr niedrig / *very low* (-2)	0.0% (0x)
niedrig / *low* (-1)	2.8% (2x)
angemessen / *appropriate* (0)	9.9% (7x)
hoch / *high* (1)	28.2% (20x)
sehr hoch / *very high* (2)	59.2% (42x)

Mean value: 1,44   **Standard dev.:** 0,78

Q-25: 1,00   **Median:** 2,00   **Q-75:** 2,00
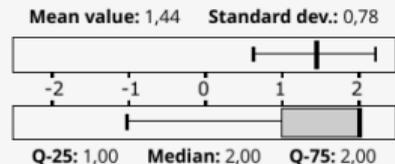
Wie beurteilen Sie die inhaltlichen Anforderungen (Schwierigkeit) dieser LV? /
How would you rate the content requirements (difficulty) of this course? (71 x answered)

sehr niedrig / *very low* (-2)	0.0% (0x)
niedrig / *low* (-1)	0.0% (0x)
angemessen / *appropriate* (0)	14.1% (10x)
hoch / *high* (1)	40.9% (29x)
sehr hoch / *very high* (2)	45.1% (32x)

Mean value: 1,31   **Standard dev.:** 0,70

Q-25: 1,00   **Median:** 1,00   **Q-75:** 2,00
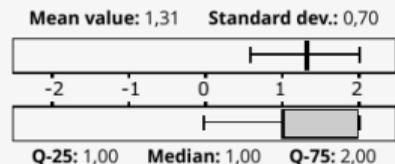
Wie zufrieden sind Sie mit der LV insgesamt? /
How satisfied are you with this course in general? (71 x answered)

sehr unzufrieden / *highly dissatisfied* (1)	**2.8%** (2x)
(2)	**11.3%** (8x)
(3)	**8.5%** (6x)
(4)	**11.3%** (8x)
(5)	**33.8%** (24x)
sehr zufrieden / *highly satisfied* (6)	**32.4%** (23x)

Mean value: 4,59   **Standard dev.:** 1,44

1 2 3 4 5 6

Q-25: 4,00   **Median:** 5,00   **Q-75:** 6,00
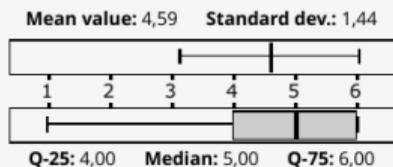
Wie stark trifft die folgende Aussage Ihrer Meinung nach auf die LV zu?:
"In dieser LV werden alle Studierenden gleichermaßen fair behandelt (unabhängig von Lerntyp, Geschlecht, Ethnie)" /
How well does the following statement describe this course, in your opinion?
"On this course, all students are treated fairly and equally (regardless of their learning style, gender and ethnicity)." (71 x answered)

trifft überhaupt nicht zu / *not at all well* (1)	**2.8%** (2x)
(2)	**2.8%** (2x)
(3)	**1.4%** (1x)
(4)	**4.2%** (3x)
(5)	**19.7%** (14x)
trifft völlig zu / *very well* (6)	**69.0%** (49x)

Mean value: 5,42   **Standard dev.:** 1,15

1 2 3 4 5 6

Q-25: 5,00   **Median:** 6,00   **Q-75:** 6,00
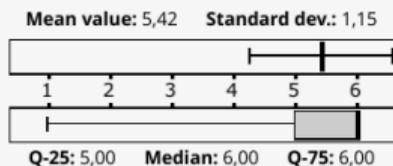
I was well prepared for this course from previous courses. (70 x answered)

trifft zu (3) 5 Comm.		**32.9%** (23x)
trifft eher zu (2) 10 Comm.		**45.7%** (32x)
trifft wenig zu (1) 4 Comm.	**15.7%** (11x)	
trifft nicht zu (0) 3 Comm.	**5.7%** (4x)	

Mean value: 2,06   Standard dev.: 0,84

Q-25: 2,00   Median: 2,00   Q-75: 3,00
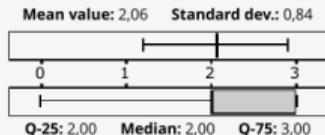
Comments on "trifft zu"

- Nur SLP konnte hier wirklich als Vorbereitung angesehen werden, da alle sonstigen LVen eigentlich nicht viel mit Low-Level Development zu tun hatten.
- Vor allem SLP
- SLP
- As a somewhat unrelated point I'd just like to say that I think prof. Gruss puts way too much stock into A0 as an indicator of preparedness for SLP. From what I've seen actual grades at SLP and OS haven't really gotten worse in recent times.
- SLP bereitet einen perfekt vor

Comments on "trifft eher zu"

- SLP für locking, Threads und shared Memory und C++
- I would say i was prepared, but of course a lot of new topics
- SLP ja
- Der einzige Kurs, welcher als Vorbereitung anzusehen ist, ist SLP. Die Kurse davor haben meines Erachtens sehr wenig für OS beigetragen.
- schon eher gut vorbereitet, durch SLP aber könnte besser sein
- slp war enorme Hilfe
- Durch SLP kriegt man schon eine gute Vorbereitung, doch es wart trotzdem recht schwer.
- I lacked in: basics like good git-handling in a group, good use of debugging and asserts, etc
- SLP, OOP und ESP prinzipiell sehr Hilfreich. Aber die LV war am anfang schon etwas überwältigend
- In SLP könnte mehr Perspektive gegeben werden, wie genau die Inhalte in OS dann gebraucht werden einfach ein kleiner Spoiler, dass man leichter das große Ganze versteht.

Der Stil der Abgabegespräche mit meinem Tutor war (70 x answered)

sehr angenehm (3) 5 Comm.	84.3% (59x)	
ok (2) 3 Comm.	14.3% (10x)	
unangenehm wegen (1)	0.0% (0x)	
eine Qual weil (0) 1 Comm.	1.4% (1x)	
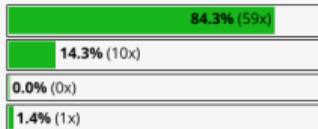
Mean value: 2,81   Standard dev.: 0,49

0   1   2   3

Q-25: 3,00   Median: 3,00   Q-75: 3,00
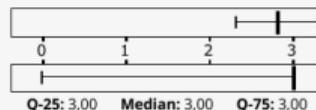
Comments on "sehr angenehm"

- our tutor did not make a big deal out of the assignment interview and it did help me feel more relaxed, even though i shat my pants every times before it. in the interview after talking about our code our tutor said ok great now only the coding example and then youre done. i really liked how they made the coding example out to be very easy and definitely doable, it took the stress away a bit. of course out tutor graded us accordingly afterwards and gave deductions for mistakes, so they didnt just give away free points.
- -
- not to difficult or stressful and the tutor really wanted to see our understanding of the os, which seems like the best approach
- Da er ein sehr gemütlicher Typ war, waren dementsprechend auch seine Abgabegespräche, sofern man auch mitgearbeitet hat.
- Even as the tutor was absolutely eviscerating us in A1 going through RC after RC it felt like a great learning experience. Both tutors gave us chance to argue for more points and we actually got a few in A2. My only complaint is that I think some of the A2 AG tasks my teammates got were too difficult but I wrote about that elsewhere. Additionally, I really appreciated that we got a git patch with comments highlighting the mistakes we made in the first assignment.

Gab es Lehrveranstaltungen, die Ihnen als besonders POSITIV in Erinnerung geblieben sind? Wenn ja, benennen Sie diese bitte im Textfeld und begründen Sie Ihre Antwort. (18 x answered)

ja 15 Comm.	**83.3%** (15x)
nein	**16.7%** (3x)

Comments on "ja"

- OS
- Betriebssysteme. Es war zwar super schwierig, aber in keinem anderen Fach hab ich so viel gelernt und auch nichts so spannendes.
- Objektorientiere Programmierung 2, Projektmanagement, Agile Software Development
- InfoSec
- 373.336 Bürgerliches Recht und Unternehmensrecht / 373.314 Betriebssoziologie; Sehr spannende Aufbereitung des Stoffes mit vielen praxisnahen Beispielen; 716.062 Softwareentwickung und Wissensmanagement; Interessante Lv-Organisation
- InfoSec, CompStat, Web Technology
- Operating Systems - super organisiert und man lernt extrem viel - auch wenn es sehr aufwendig ist
- Operating Systems
- Information Security
- Operating Systems, System Level Programming, Objektorientierte Programmierung 1 & 2, Computer Grafik & Vision - Mit Abstand die interessantesten Fächer meines Bachelorstudiums. Diese hatten das vermittelte Wissen, das man sich in sehr vielen (anderen) Fächern angeeignet hatte, am besten in die Praxis umgesetzt mit sehr sehr interessanten Projekten!
- Bachelorarbeit und Operating systems
- Auch wenn man sehr gefordert wurde, war Betriebssysteme doch eine sehr gute LV.
- Fundamentals of Geometry Processing
- Ja, OS, weil super präsentiert. FoCS, Prof Slany hat einen super Vortragsstil für das erste Semester.
- System Level Programming, Rechnernetze und Organisation, Software Architecture

Vielleicht hätte man hervorheben sollen, dass uSTL NICHT kompatibel zu STL ist!

Daniel Gruss

Vielleicht hätte man hervorheben sollen, dass uSTL NICHT kompatibel zu STL ist!
ustl::list gehört eigentlich verboten.

Vielleicht hätte man hervorheben sollen, dass uSTL NICHT kompatibel zu STL ist!
ustl::list gehört eigentlich verboten. Man kommt in Teufels Küche, wenn man von ihr
ableiten möchte.

Vielleicht hätte man hervorheben sollen, dass uSTL NICHT kompatibel zu STL ist! ustl::list gehört eigentlich verboten. Man kommt in Teufels Küche, wenn man von ihr ableiten möchte. Man gibt sich einer Illusion hin, die nicht stimmt.

Vielleicht hätte man hervorheben sollen, dass uSTL NICHT kompatibel zu STL ist! ustl::list gehört eigentlich verboten. Man kommt in Teufels Küche, wenn man von ihr ableiten möchte. Man gibt sich einer Illusion hin, die nicht stimmt. Einfügen und Löschen sind NICHT effizient und Iteratoren verlieren die Gültigkeit, wo sie es nicht sollten.

Vielleicht hätte man hervorheben sollen, dass uSTL NICHT kompatibel zu STL ist!
ustl::list gehört eigentlich verboten. Man kommt in Teufels Küche, wenn man von ihr
ableiten möchte. Man gibt sich einer Illusion hin, die nicht stimmt. Einfügen und
Löschen sind NICHT effizient und Iteratoren verlieren die Gültigkeit, wo sie es nicht
sollten. Trotzdem war es hilfreich.

Check the standard!

```
iterator erase (iterator position);
iterator erase (iterator first, iterator last);
```

erase verwenden ohne den iterator upzudaten geht mit der GNU C++ STL *meistens...*
→ gefährlich sich anzugewöhnen den iterator zu ignorieren

Conclusion

To conclude, we can get some facts about each data structure:

- std::list is very very slow to iterate through the collection due to its poor spatial locality.
- std::vector and std::deque perform always faster than std::list with very small data
- std::list handles very well large elements
- std::deque performs better than a std::vector for inserting at random positions (especially at the front, which is constant time)
- std::deque and std::vector do not support very well data types with high cost of copy/assignment

This draw simple conclusions on usage of each data structure:

- Number crunching: use std::vector or std::deque
- Linear search: use std::vector or std::deque
- Random Insert/Remove:
  - Small data size: use std::vector
  - Large element size: use std::list (unless if intended principally for searching)
- Non-trivial data type: use std::list unless you need the container especially for searching. But for multiple modifications of the container, it will be very slow.
- Push to front: use std::deque or std::list

http://baptiste-wicht.com/posts/2012/12/
cpp-benchmark-vector-list-deque.html

- We try to improve our support constantly
- Feedback, Evaluations

"We will not lower the bar, but we will do what we can to help you over it."

Daniel Gruss

- Bad time management
- Pair programming (it's not efficient enough!!)
- Problems with working in a team
- No C/C++ experience
- Not willing to learn and use C/C++

You and SWEB

Organizational Details

SWEB

Booting SWEB

Back to Assignment 1

Daniel Gruss

Channels

- Website: `https://iaik.tugraz.at/os`

Daniel Gruss

- Website: `https://iaik.tugraz.at/os`
- Discord (tag your tutor, also in your own group channels!)
- Email: os@iaik.tugraz.at
- Consultation hour: send us an email
- *During the question days: slower response times on Discord/email ;)*

Daniel Gruss

Student Debates

- Take place before the design deadlines
- You are expected to bring a sketch/summary of your design
- You are expected to have done proof of concept implementations by then

Daniel Gruss

Student Debates

- Take place before the design deadlines
- You are expected to bring a sketch/summary of your design
- You are expected to have done proof of concept implementations by then
- Prepare to defend your ideas

Daniel Gruss

Design-PoC

- Only code! No design document!

- Proof of concept implementation

- You get up to 5 points, depending on the test system score of your submission.

Daniel Gruss

Git is mandatory

- You have to:
  - push regularly into the provided repository
  - use your real name and email address for commits

Daniel Gruss

- Personal maximum number of points

- Personal maximum number of points
- Unlock points:
  - For each commit you unlock 0.5 points
  - For each 10 LoC added you unlock 0.5 points
  - Max. 10 points per day can be unlocked

- Personal maximum number of points
- Unlock points:
  - For each commit you unlock 0.5 points
  - For each 10 LoC added you unlock 0.5 points
  - Max. 10 points per day can be unlocked
- Average: > 55 points unlocked
- Test system shows your current personal maximum number of points

Daniel Gruss

- We even give you master access to your repository, but:

- We even give you master access to your repository, but:
  - renaming repo = your group fails the course
  - changing repo path = your group fails the course
  - adding/removing group members = your group fails the course

Every team member must understand all parts and contribute points to the mandatory task categories!

Daniel Gruss

Daniel Gruss

Daniel Gruss
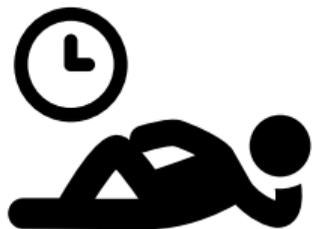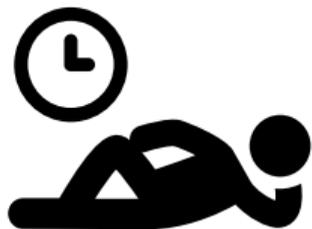
- Set internal deadlines for your team members

- Set internal deadlines for your team members
    - No idea how to split the work and choose good internal deadlines? Ask tutor for recommendations!

Daniel Gruss

- Set internal deadlines for your team members
  - No idea how to split the work and choose good internal deadlines? Ask tutor for recommendations!
- Missed deadline? $\rightarrow$ re-assign task

Daniel Gruss
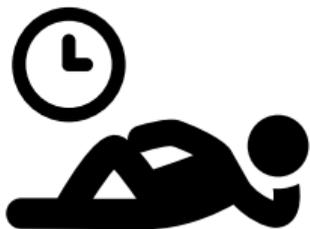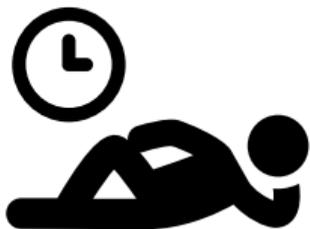
- Set internal deadlines for your team members
    - No idea how to split the work and choose good internal deadlines? Ask tutor for recommendations!
- Missed deadline? $\rightarrow$ re-assign task
- Repeatedly missed deadline $\rightarrow$ tell tutor to remove team member from team

Daniel Gruss

Daniel Gruss

- Team member(s) not contributing enough $\rightarrow$ contact tutor via mail!

Daniel Gruss

What if you as a team don't fit together?

- Team member(s) not contributing enough $\rightarrow$ contact tutor via mail!
- There will be more students with the same problem as you have

Daniel Gruss

What if you as a team don't fit together?

- Team member(s) not contributing enough → contact tutor via mail!
- There will be more students with the same problem as you have
- Team doesn't want to split but you want to leave? → also possible

Daniel Gruss

What if you as a team don't fit together?

- Team member(s) not contributing enough $\rightarrow$ contact tutor via mail!
- There will be more students with the same problem as you have
- Team doesn't want to split but you want to leave? $\rightarrow$ also possible
- $\rightarrow$ Merges are possible

- Team member(s) not contributing enough $\rightarrow$ contact tutor via mail!
- There will be more students with the same problem as you have
- Team doesn't want to split but you want to leave? $\rightarrow$ also possible
$\rightarrow$ Merges are possible
$\rightarrow$ Don't pull anyone through

Daniel Gruss

What if you as a team don't fit together?

- Team member(s) not contributing enough $\rightarrow$ contact tutor via mail!
- There will be more students with the same problem as you have
- Team doesn't want to split but you want to leave? $\rightarrow$ also possible
- $\rightarrow$ Merges are possible
- $\rightarrow$ Don't pull anyone through
- small number of bonus points as compensation (if **we** think it's appropriate)

Daniel Gruss

What if you as a team don't fit together?

- Team member(s) not contributing enough $\rightarrow$ contact tutor via mail!
- There will be more students with the same problem as you have
- Team doesn't want to split but you want to leave? $\rightarrow$ also possible
- $\rightarrow$ Merges are possible
- $\rightarrow$ Don't pull anyone through
- small number of bonus points as compensation (if **we** think it's appropriate)

Daniel Gruss

What if you as a team don't fit together?

- Team member(s) not contributing enough → contact tutor via mail!
- There will be more students with the same problem as you have
- Team doesn't want to split but you want to leave? → also possible
- → Merges are possible
- → Don't pull anyone through
- small number of bonus points as compensation (if **we** think it's appropriate)

Not happy with your tutor's resolution?

Daniel Gruss

What if you as a team don't fit together?

- Team member(s) not contributing enough $\rightarrow$ contact tutor via mail!
- There will be more students with the same problem as you have
- Team doesn't want to split but you want to leave? $\rightarrow$ also possible
$\rightarrow$ Merges are possible
$\rightarrow$ Don't pull anyone through
- small number of bonus points as compensation (if **we** think it's appropriate)

Not happy with your tutor's resolution? $\rightarrow$ **Talk to me**, we'll figure out a solution

Daniel Gruss

- Existing syscall wrapper functions are POSIX-compatible

- Existing syscall wrapper functions are POSIX-compatible
  - Do not change signatures!
  - We have automated tests using the POSIX interface

Daniel Gruss

- Existing syscall wrapper functions are POSIX-compatible
  - Do not change signatures!
  - We have automated tests using the POSIX interface
- If you implement new functions, try to make signatures POSIX-compatible

- Writing numerous test programs is unavoidable

Testing

- Writing numerous test programs is unavoidable
- Test programs are supposed to show whether your implementation works according to the assignment
- You will get points for the test programs

Daniel Gruss

Testing

- Writing numerous test programs is unavoidable
- Test programs are supposed to show whether your implementation works according to the assignment
- You will get points for the test programs
- We have our own secret test programs
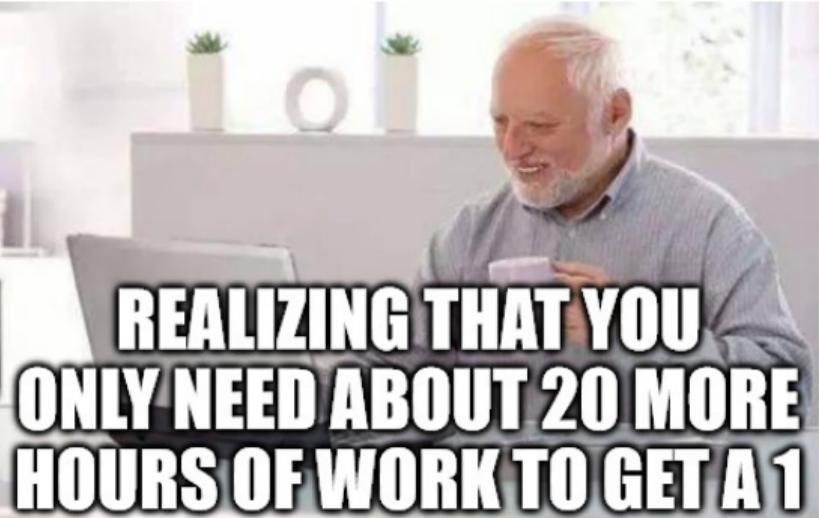
Daniel Gruss

REALIZING THAT YOU ONLY NEED ABOUT 20 MORE HOURS OF WORK TO GET A 1

IT'S 2 HOURS BEFORE THE DEADLINE

- Tag the commit you want to submit: `git tag SubmissionD1 [commit_hash]`

Submissions

- Tag the commit you want to submit: git tag SubmissionD1 [commit_hash]
- Push to repository: git push / git push --tags

Daniel Gruss

Submissions

- Tag the commit you want to submit: git tag SubmissionD1 [commit_hash]
- Push to repository: git push / git push --tags
- Read the commit ID:

```
git show SubmissionD1
commit 196bc4a704f37d7f969d27a258b513693e3b30f4
Author: Daniel Gruss <daniel.gruss@iaik.tugraz.at>
```

Test System will acknowledge your submission!

Small Fixes after the Deadline

- Small fixes $\rightarrow$ small deduction
- Do this at the same time:
    - Mail to os@iaik.tugraz.at!
    - Submit and retag your fixed version!

Daniel Gruss

Assessment A1

Reference (=100%)

- Design: 5 points
- Task 1: 20 points
- Task 2: 10 points
- Elective tasks*: 15 points (or more)

Daniel Gruss

Assessment A1

Reference (=100%)

- Design: 5 points
- Task 1: 20 points
- Task 2: 10 points
- Elective tasks*: 15 points (or more)

Minimum

- Design: 1 point
- Mandatory tasks: 15 points
- And in total: 25 points (=50%)

(*) See the list of elective tasks on the Website!

Daniel Gruss

Reference (=100%)

- Design: 5 points

- Mandatory tasks: 40 points

- Elective tasks*: 5 points (or more)

Assessment A2

■

Reference (=100%)

- Design: 5 points
- Mandatory tasks: 40 points
- Elective tasks*: 5 points (or more)

Minimum

- Design: 1 point
- Mandatory tasks: 15 points
- And in total: 25 points (=50%)

(*) See the list of elective tasks on the Website!

Daniel Gruss

Minimum requirements

- A1: 25 of 50 points

- A2: 25 of 50 points

Minimum requirements

- A1: 25 of 50 points

- A2: 25 of 50 points

Limits

- A1: max. 60 points
  - Unlimited if A2 $\geq$ 40 points

- A2: unlimited points

Assessment

- mid-term exam: 35%
- the practicals: 65%

Daniel Gruss

To pass the class, you have to acquire

- overall at least 55%

Daniel Gruss

To pass the class, you have to acquire

- overall at least 55%
- at least 50% of the possible points on written exam

Daniel Gruss

To pass the class, you have to acquire

- overall at least 55%
- at least 50% of the possible points on written exam
- at least 50% of the possible points in the practicals

Daniel Gruss

- Didn't pass any assignment or exam?

Getting a positive grade remains as easy, getting a 1 gets harder (with every second chance)

- Didn't pass any assignment or exam?
- No worries, we have a second chance for everyone.

Getting a positive grade remains as easy, getting a 1 gets harder (with every second chance)

- Didn't pass any assignment or exam?
- No worries, we have a second chance for everyone.
- DD $\rightarrow$ you can continue without Second Chance but with 0 DD/PoC points

Getting a positive grade remains as easy, getting a 1 gets harder (with every second chance)

- Didn't pass any assignment or exam?
- No worries, we have a second chance for everyone.
- DD $\rightarrow$ you can continue without Second Chance but with 0 DD/PoC points
- PoC $\rightarrow$ Second Chance PoC (examined by Lecturers)

Getting a positive grade remains as easy, getting a 1 gets harder (with every second chance)

Assessment: Second Chance

- Didn't pass any assignment or exam?
- No worries, we have a second chance for everyone.
- DD $\rightarrow$ you can continue without Second Chance but with 0 DD/PoC points
- PoC $\rightarrow$ Second Chance PoC (examined by Lecturers)
- Exercise $\rightarrow$ Second Chance Exercise (examined by Tutors + Lecturers)

Getting a positive grade remains as easy, getting a 1 gets harder (with every second chance)

Daniel Gruss

Assessment: Second Chance

- Didn't pass any assignment or exam?
- No worries, we have a second chance for everyone.
- DD $\rightarrow$ you can continue without Second Chance but with 0 DD/PoC points
- PoC $\rightarrow$ Second Chance PoC (examined by Lecturers)
- Exercise $\rightarrow$ Second Chance Exercise (examined by Tutors + Lecturers)
- Exam $\rightarrow$ Second Chance Exam

Getting a positive grade remains as easy, getting a 1 gets harder (with every second chance)

Daniel Gruss

Points

$\rightarrow$ final points for assignment

Daniel Gruss

Points

1. Take the team score

$\rightarrow$ final points for assignment

Daniel Gruss

Points

1. Take the team score
2. cap it by the personal unlocked points

$\rightarrow$ final points for assignment

Daniel Gruss

Points

1. Take the team score

2. cap it by the personal unlocked points

3. apply DD deduction

$\rightarrow$ final points for assignment

Points

1. Take the team score

2. cap it by the personal unlocked points

3. apply DD deduction

4. apply AG deduction

$\rightarrow$ final points for assignment

Points (Example)

Daniel Gruss

Points (Example)

1. You get 48 points as a team $\rightarrow$ 48

Points (Example)

1. You get 48 points as a team $\rightarrow$ 48

2. Your personal unlocked points are 43 $\rightarrow$ 43

Daniel Gruss

Points (Example)

1. You get 48 points as a team $\rightarrow$ 48
2. Your personal unlocked points are 43 $\rightarrow$ 43
3. Your group had 4 points on the PoC but you didn't say enough in the DD $\rightarrow$ 39

Daniel Gruss

Points (Example)

1. You get 48 points as a team → 48
2. Your personal unlocked points are 43 → 43
3. Your group had 4 points on the PoC but you didn't say enough in the DD → 39
4. You get a 75% deduction because you couldn't really figure out where to do what in your SWEB in the AG → 10 points → second chance

Daniel Gruss

Points (Example)

Daniel Gruss

Points (Example)

1. You get 48 points as a team $\rightarrow$ 48

Points (Example)

1. You get 48 points as a team $\rightarrow$ 48

2. Your personal unlocked points are 57 $\rightarrow$ 48

Daniel Gruss

Points (Example)

1. You get 48 points as a team $\rightarrow$ 48

2. Your personal unlocked points are 57 $\rightarrow$ 48

3. You have a 25% deduction because you didn't say enough in the DD $\rightarrow$ 36

Points (Example)

1. You get 48 points as a team → 48
2. Your personal unlocked points are 57 → 48
3. You have a 25% deduction because you didn't say enough in the DD → 36
4. You get a 20% deduction because you took too long with making changes → 29
   → barely positive

Daniel Gruss

Points (Example)

Daniel Gruss
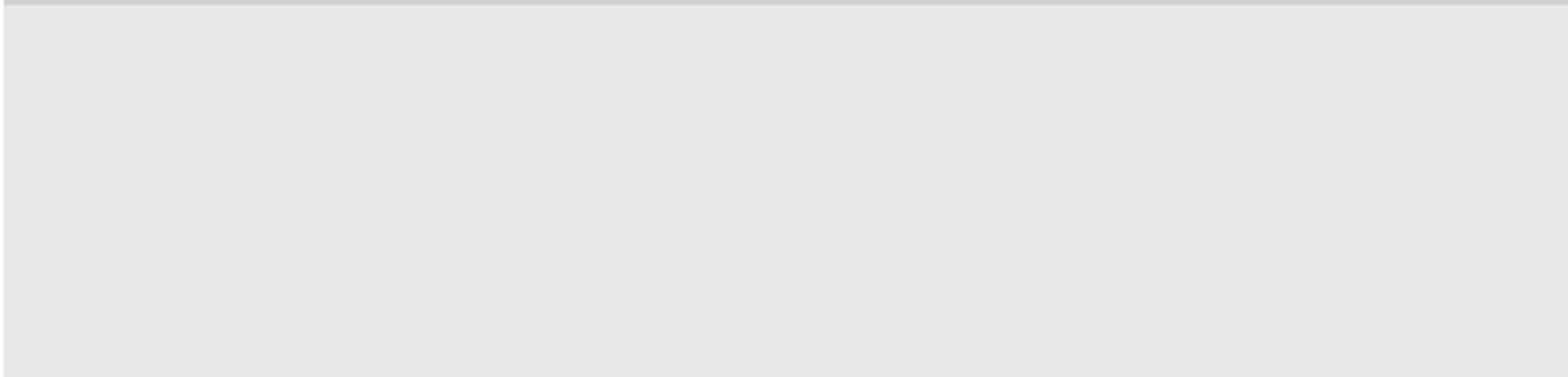
Points (Example)

1. You get 45 points as a team → 45

Points (Example)

1. You get 45 points as a team $\rightarrow$ 45

2. Your personal unlocked points are 70 $\rightarrow$ 45

Points (Example)

1. You get 45 points as a team $\rightarrow$ 45
2. Your personal unlocked points are 70 $\rightarrow$ 45
3. You participated in the DD properly $\rightarrow$ 45

Points (Example)

1. You get 45 points as a team $\rightarrow$ 45
2. Your personal unlocked points are 70 $\rightarrow$ 45
3. You participated in the DD properly $\rightarrow$ 45
4. You get a 5% deduction because you took rather long with making changes $\rightarrow$ 43
   $\rightarrow$ good grade

Daniel Gruss

Marks

- genügend: 55-66
- befriedigend: 67-78
- gut: 79-89
- sehr gut: 90+

Daniel Gruss

Bonus points

We want you to evaluate. From 30% upwards: for every 10 percentage points, **everyone** needs 1 point less for a better positive grade!

Daniel Gruss

Review meetings (Abgabegespräche)

As a group

- You explain what you implemented and how you tested it
- Together with the teaching assistant you determine the points of your group

As a group member

- You are able to read, explain and change the code
- You can implement small new features or extend existing ones
  - just quickly, no rigorous testcase writing
  - tutor will stop you as soon as it's clear that you deserve the points
- Otherwise you will get less points

Daniel Gruss

Plagiarism

- Discussions with other teams are appreciated
- But: no collaboration!

Daniel Gruss

- Discussions with other teams are appreciated

- But: no collaboration!

- We check for plagiarism

- Similarities $\rightarrow$ teams are questioned

Plagiarism

- Discussions with other teams are appreciated
- But: no collaboration!
- We check for plagiarism
- Similarities → teams are questioned
- Both teams: 0 points in either case
- At least one team: "Ungültig/Täuschung" (with all its consequences)

Daniel Gruss

Plagiarism

- Do not provide your source code to other teams

- Do not provide your source code to other teams
- Make sure your source code is protected against unintended access from others

Plagiarism

- Do not provide your source code to other teams
- Make sure your source code is protected against unintended access from others
- Do not use source code from previous years
  - Code from another team $\rightarrow$ plagiarism

Daniel Gruss

- Do not provide your source code to other teams
- Make sure your source code is protected against unintended access from others
- Do not use source code from previous years
  - Code from another team $\rightarrow$ plagiarism
  - Your own code (not exactly the same team)
    $\rightarrow$ not allowed

Plagiarism

- Do not provide your source code to other teams
- Make sure your source code is protected against unintended access from others
- Do not use source code from previous years
  - Code from another team $\rightarrow$ plagiarism
  - Your own code (not exactly the same team)
    $\rightarrow$ not allowed
  - Your own code (exactly the same team)
    $\rightarrow$ allowed, probably not the best idea ;)

Daniel Gruss

Questions so far?

Daniel Gruss

- Tutors want to help you.

Daniel Gruss

- Tutors want to help you.
- Tutors have a competition:

- Tutors want to help you.
- Tutors have a competition:
  - The higher their students score **on average**, the better they score!

Daniel Gruss

- Tutors want to help you.
- Tutors have a competition:
  - The higher their students score **on average**, the better they score!
  $\rightarrow$ they want to win, they want to make you pass, help them!

- Tutors want to help you.
- Tutors have a competition:
    - The higher their students score **on average**, the better they score!
  $\rightarrow$ they want to win, they want to make you pass, help them!
- You don't know whether you have enough points yet? Don't ask the tutor for your points, but ask them:
  "should we do more? would you recommend that we still do something? what would you recommend we do?"

Daniel Gruss

- VU Amsterdam: Minix (1987), Minix3 (2005)

Daniel Gruss

History and the others

- VU Amsterdam: Minix (1987), Minix3 (2005)
- Berkeley: Nachos (1992)

Daniel Gruss

History and the others

- VU Amsterdam: Minix (1987), Minix3 (2005)
- Berkeley: Nachos (1992)
- Stanford: Pintos (2004)

Daniel Gruss

History and the others

- VU Amsterdam: Minix (1987), Minix3 (2005)
- Berkeley: Nachos (1992)
- Stanford: Pintos (2004)
- Graz: SWEB (2006)
  - BS KU W04

Daniel Gruss

History and the others

- VU Amsterdam: Minix (1987), Minix3 (2005)
- Berkeley: Nachos (1992)
- Stanford: Pintos (2004)
- Graz: SWEB (2006)
    - BS KU W04
    - Philip Lawatsch, Bernhard Tittelbach, many other students and projects

Daniel Gruss

- VU Amsterdam: Minix (1987), Minix3 (2005)
- Berkeley: Nachos (1992)
- Stanford: Pintos (2004)
- Graz: SWEB (2006)
    - BS KU W04
    - Philip Lawatsch, Bernhard Tittelbach, many other students and projects

- Minimalistic operating system kernel
- Runs on x86-32, x86-64, ARM
- Emulated using qemu

- Minimalistic operating system kernel
- Runs on x86-32, x86-64, ARM
- Emulated using qemu
- Important features are missing
- Your task: Make your SWEB a beautiful, feature-rich kernel

Daniel Gruss

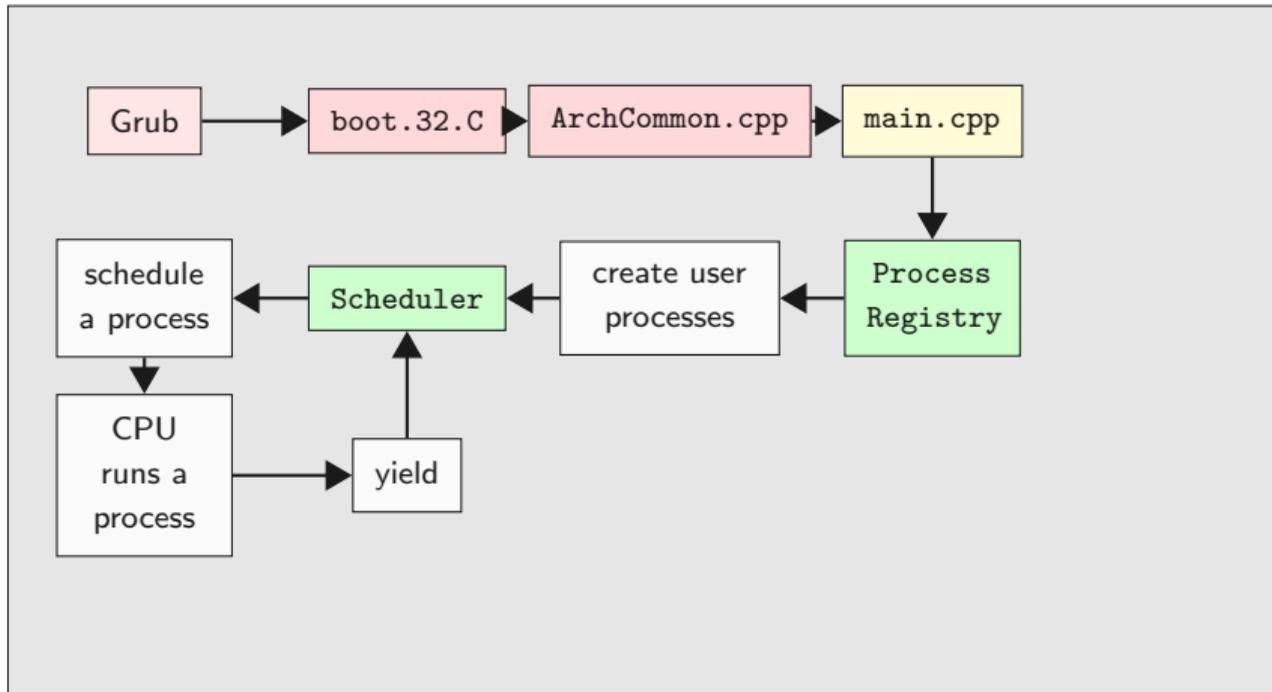Grub

Daniel Gruss

Overview

Daniel Gruss

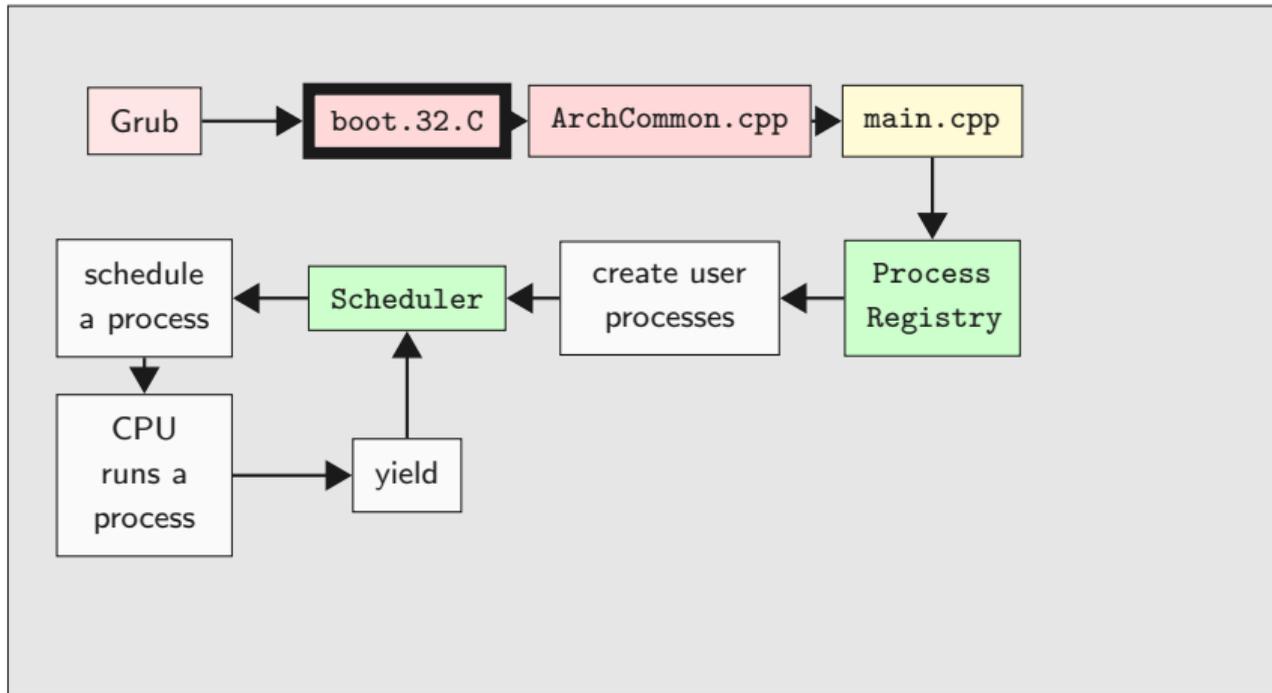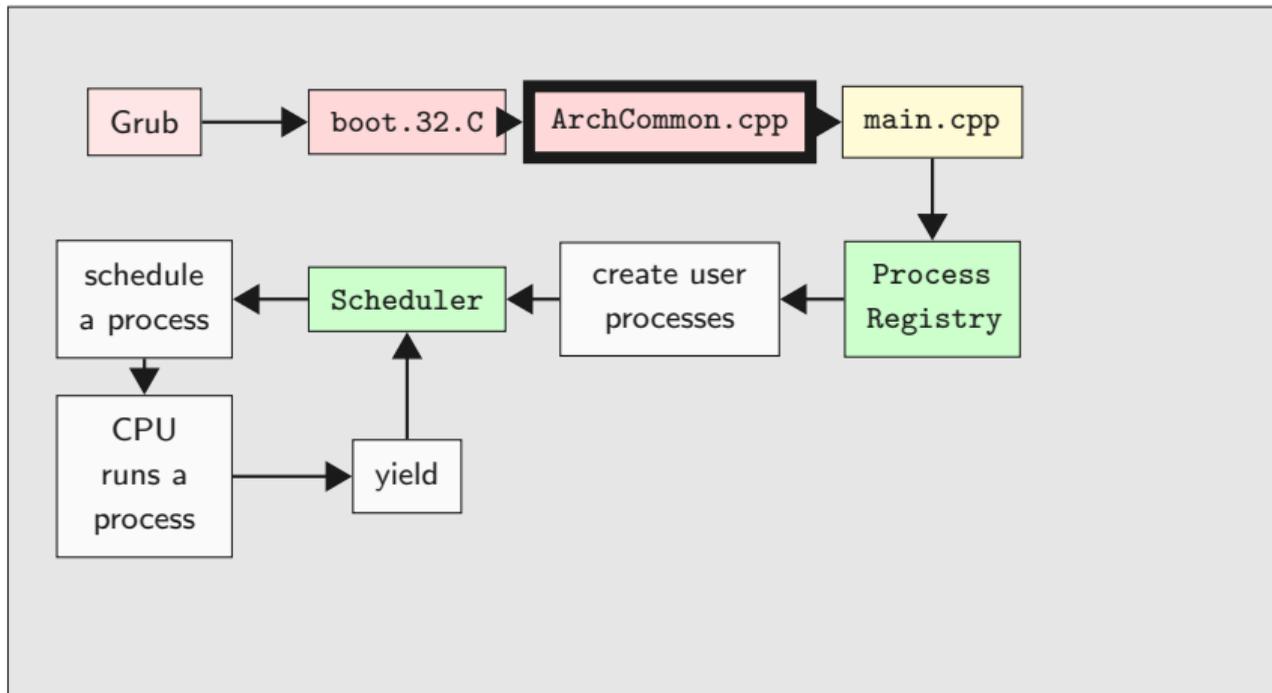Daniel Gruss

Daniel Gruss
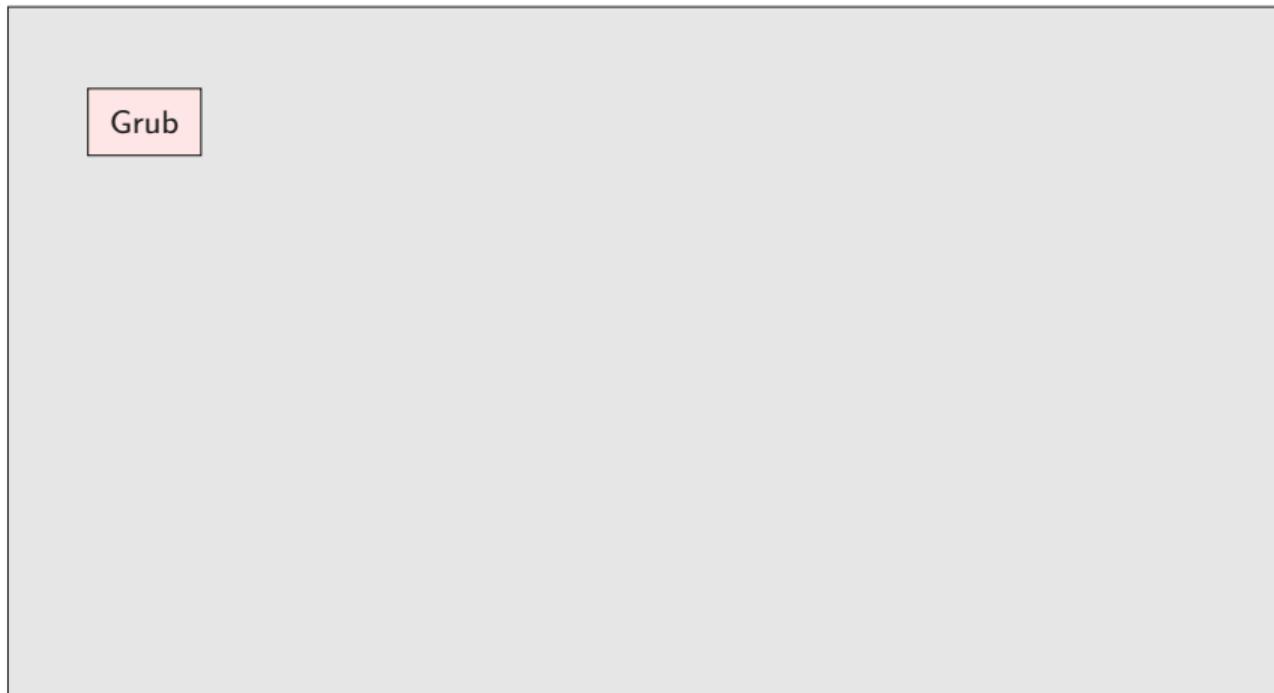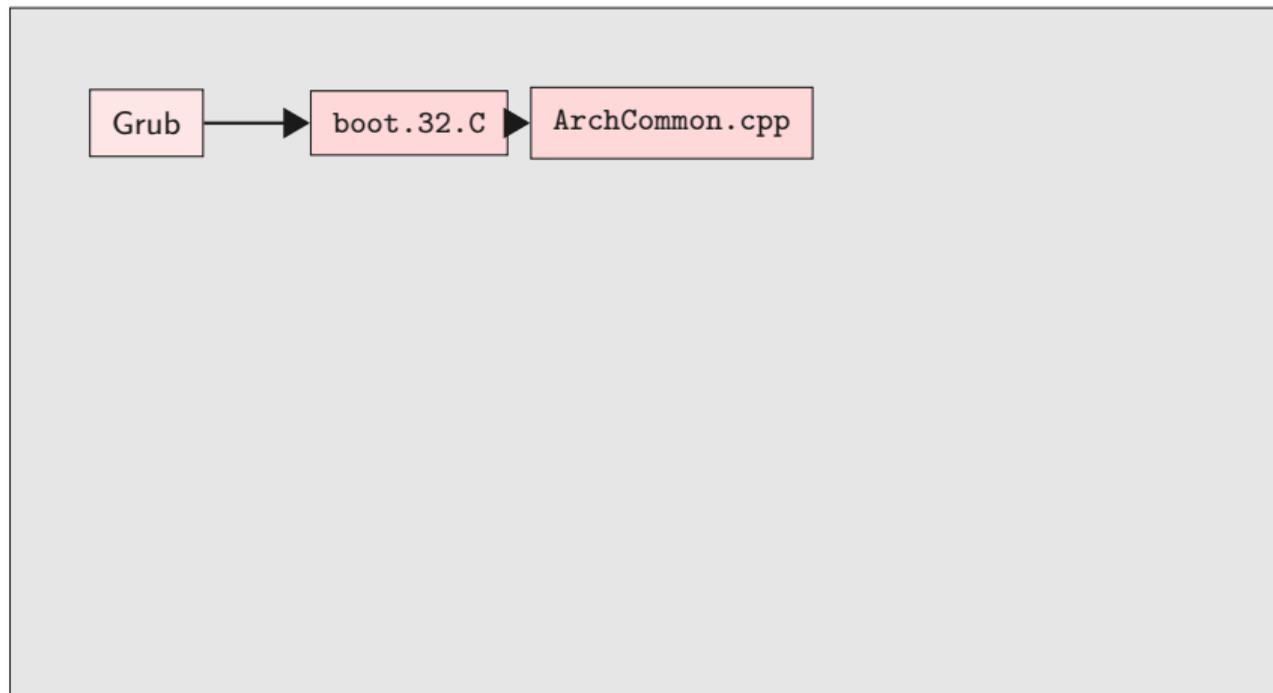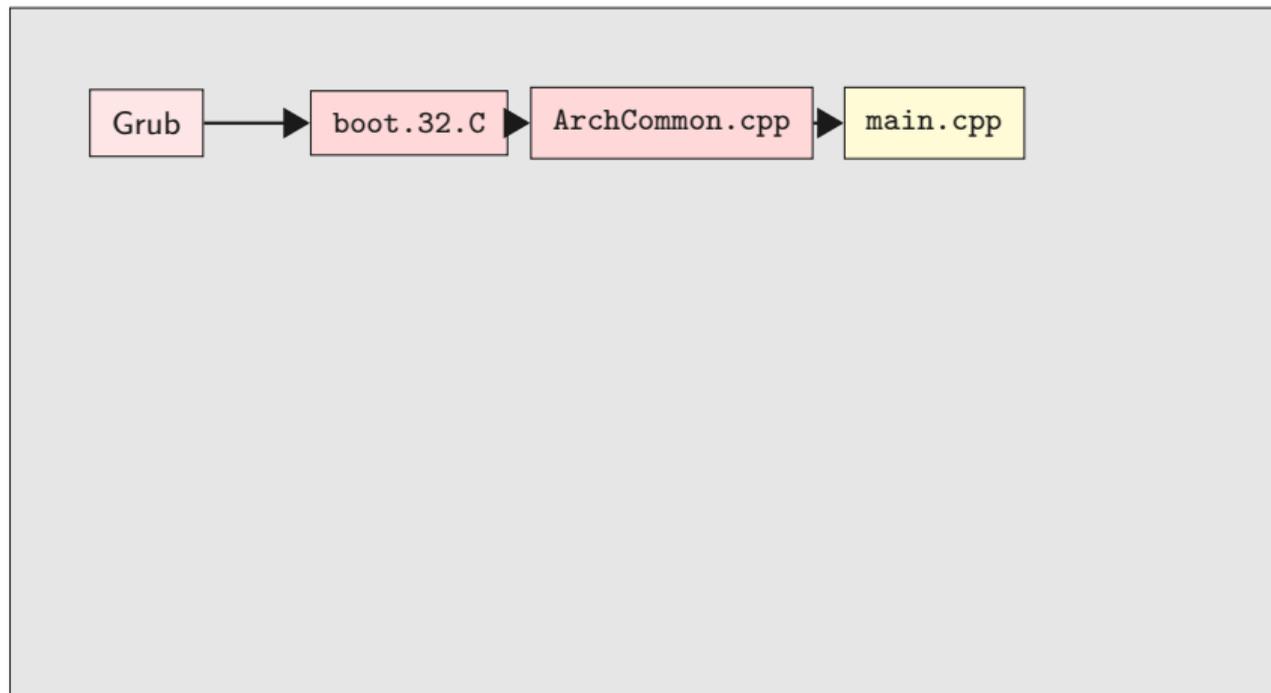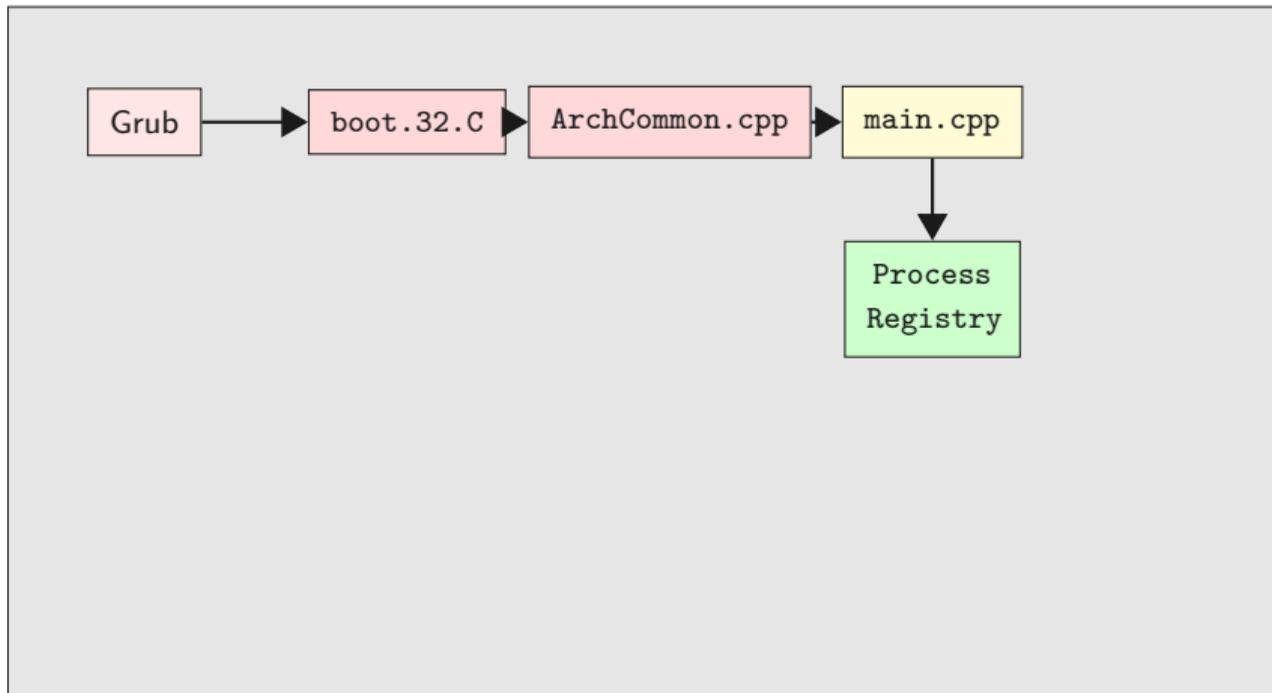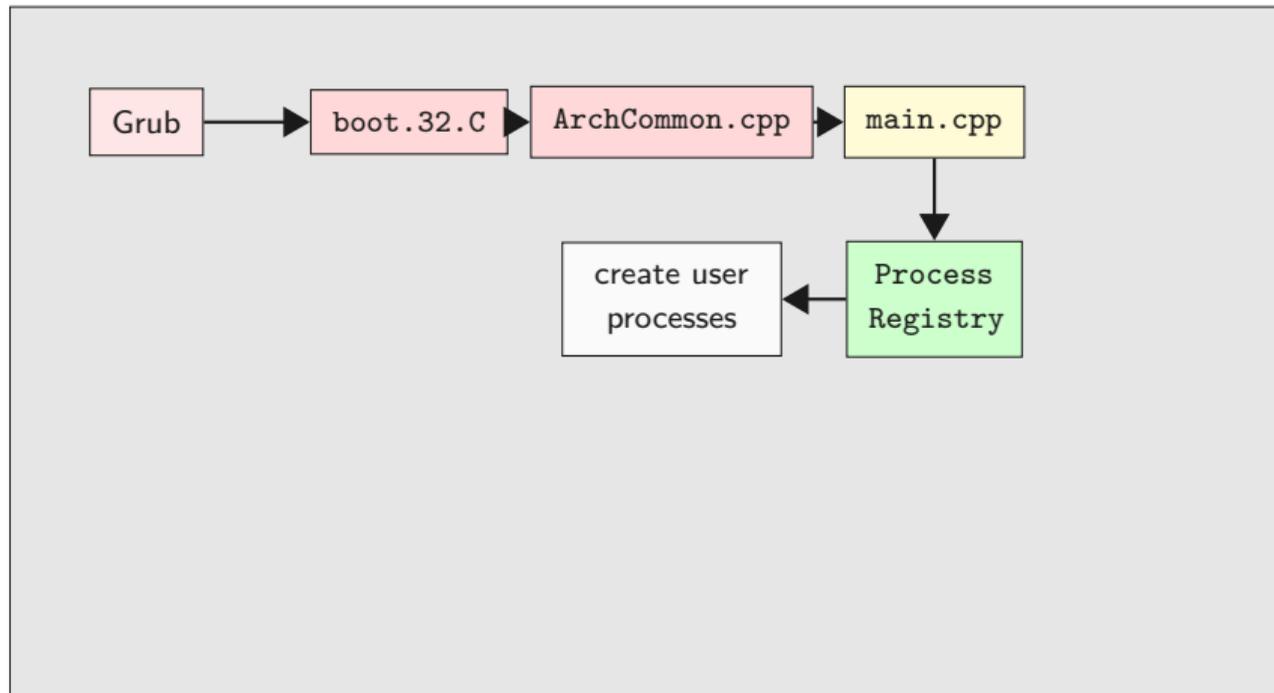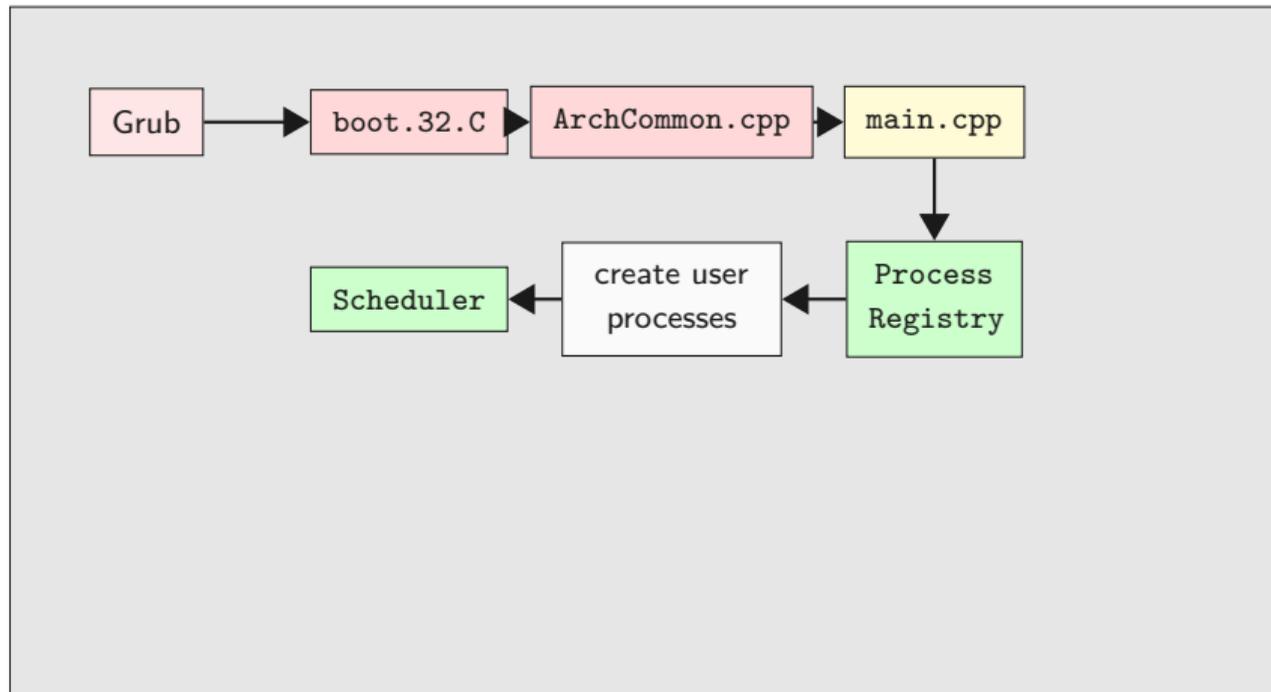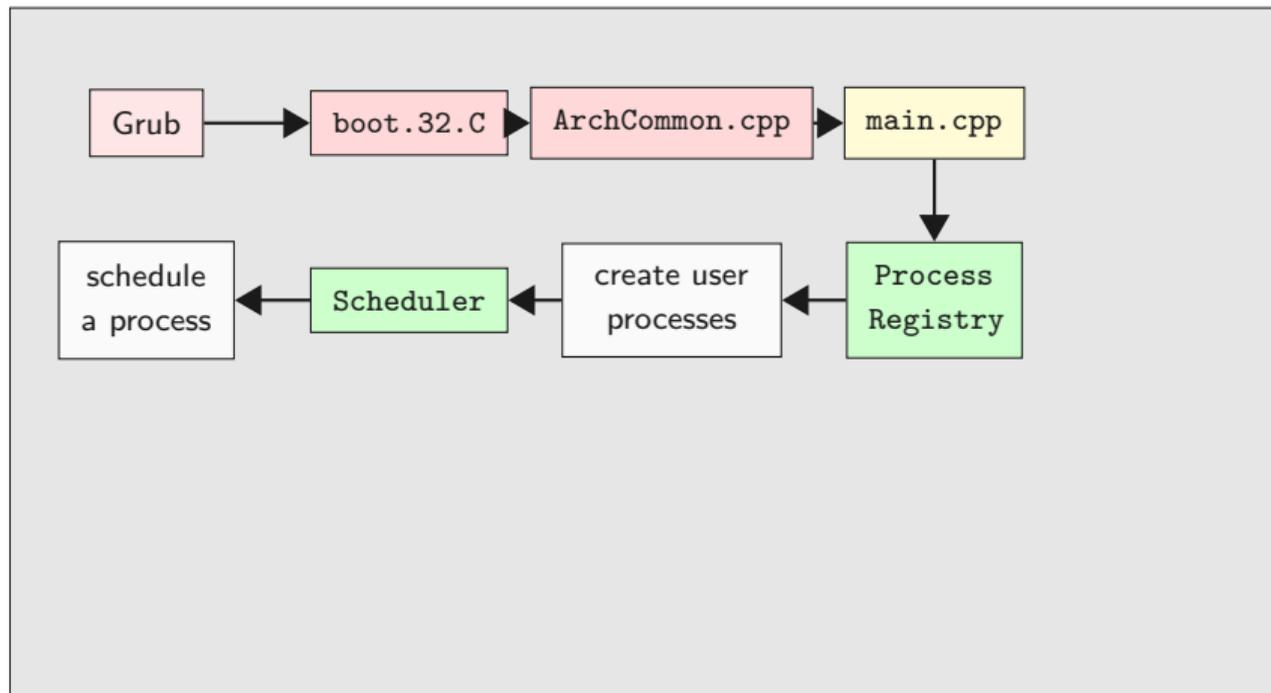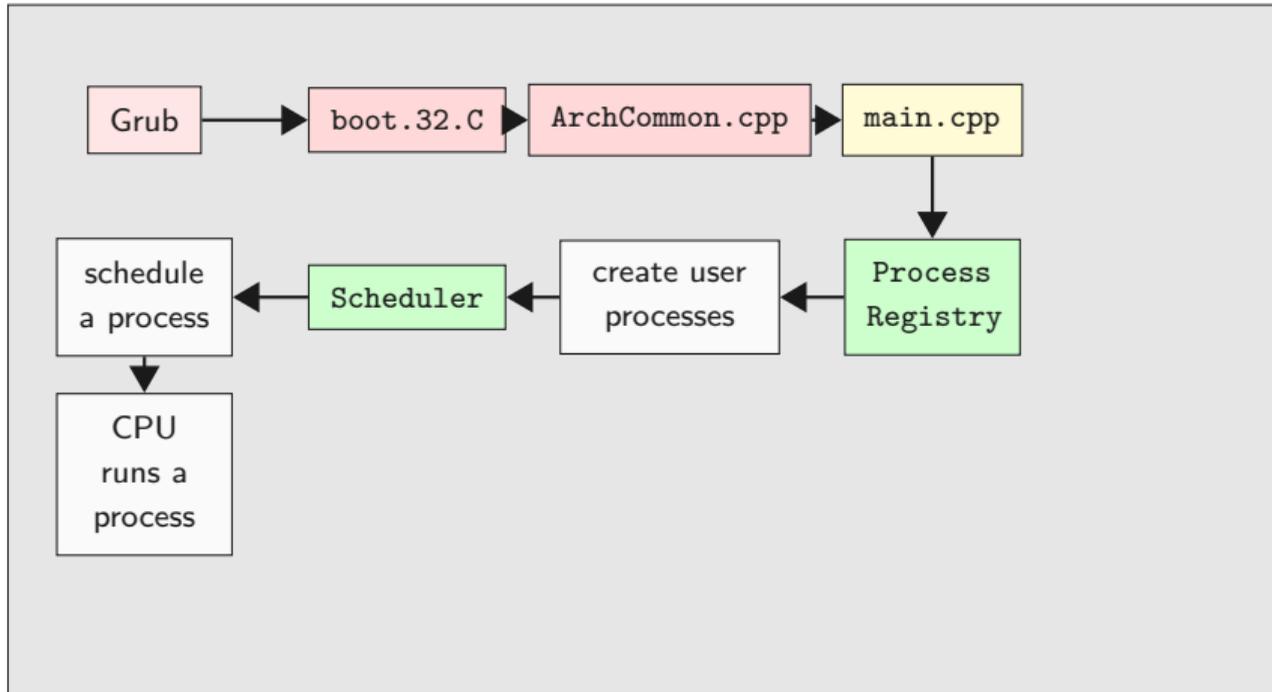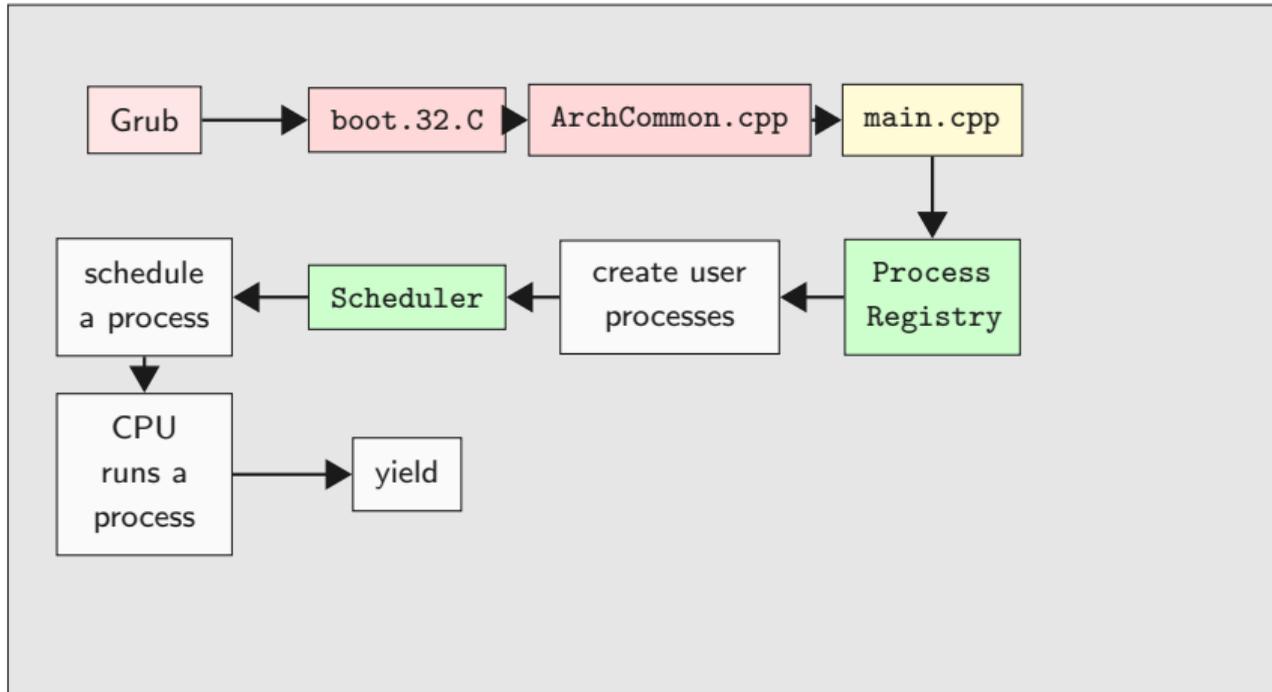
```
extern "C" void entry()
{
  PRINT("Booting...\n");
  // ...
  PRINT("Initialize Kernel Paging Structs\n");
  // ...
  PRINT("Enable Paging...\n");
  // ...
  PRINT("Calling entry64()...\n");
  asm("ljmp %[cs],$entry64-BASE\n" : : [cs]"i"(KERNEL_CS));
```

Daniel Gruss

- 32 bit
- Works on physical addresses
- Setup hardware
- Setup paging
- Jump into sane, virtual C world

Daniel Gruss

```cpp
extern "C" void entry64()
{
  // ...
  PRINT("Switch to our own stack...\n");
  asm("mov %[stack], %%rsp\n"
      "mov %[stack], %%rbp\n" : : [stack]"i"(boot_stack + 0x4000));
  PRINT("Loading Long Mode Segments...\n");
  // ...
  PRINT("Calling startup()...\n");
  asm("jmp *%[startup]" : : [startup]"r"(startup));
```
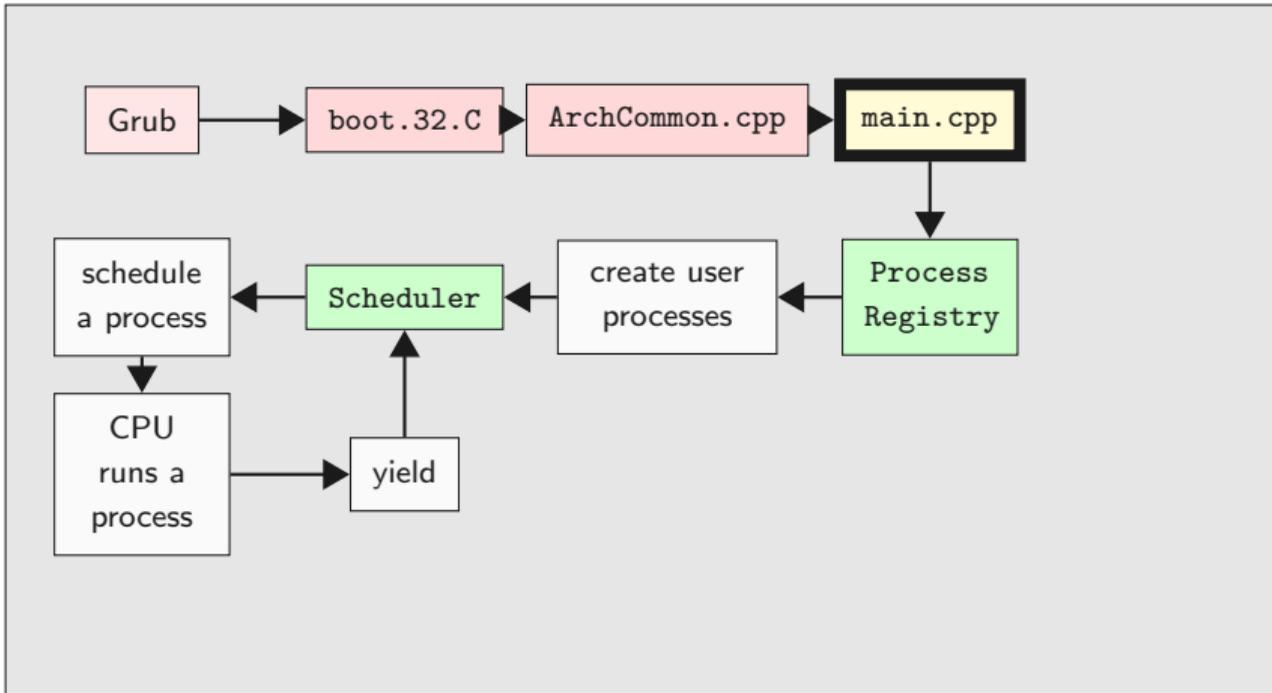
```cpp
extern "C" void startup() {
  removeBootTimeIdentMapping();
  system_state = BOOTING;
  PageManager::instance();
  writeLine("PageManager and KernelMemoryManager created\n");
  // ...
  Scheduler::instance()->addNewThread(new ProcessRegistry(...));
  // ...
  system_state = RUNNING;
  ArchInterrupts::enableInterrupts();
  Scheduler::instance()->yield();
  //not reached
```

- Setup kernel objects
- Setup more hardware
- Enable interrupts
- Handover control to `Scheduler`

Daniel Gruss
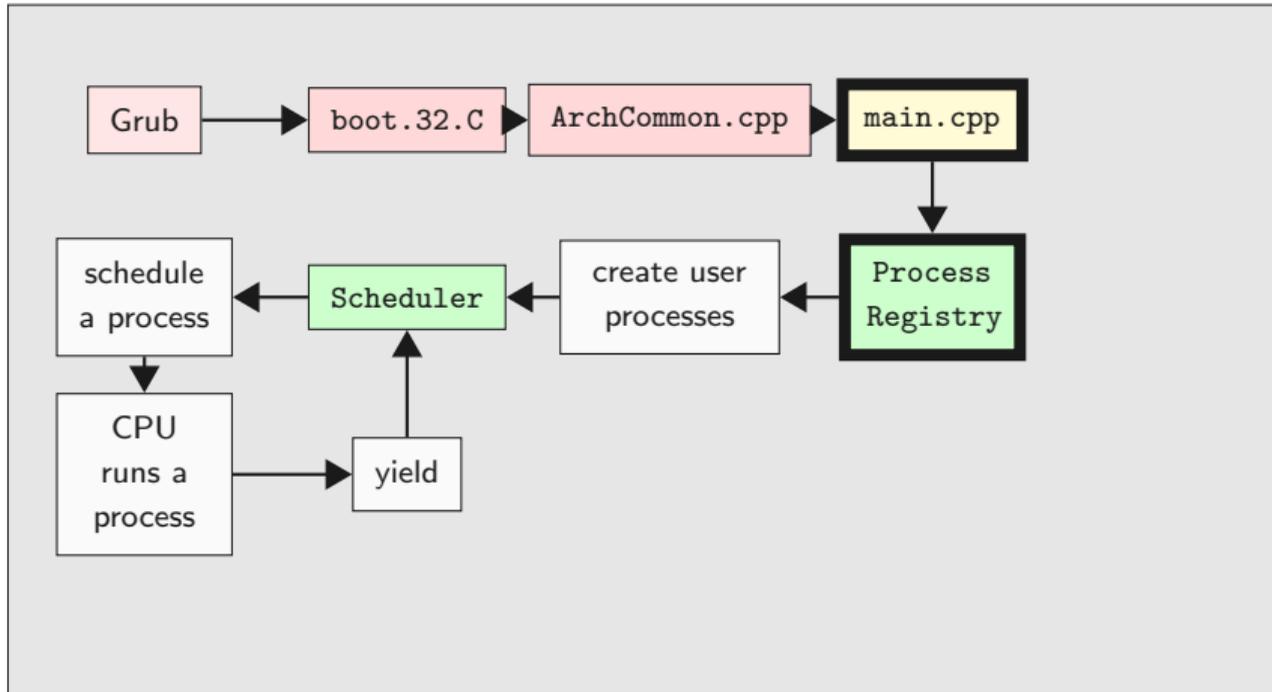
```cpp
void ProcessRegistry::Run()
{
  debug(PROCESS_REG, "mounting userprog-partition \n");
  // ...
  vfs_syscall.mount("idea1", "/usr", "minixfs", 0);
  debug(PROCESS_REG, "mount idea1\n");

  for (size_t i = 0; progs_[i]; i++)
  {
    createProcess(progs_[i]);
  }
```

- Kernel Thread
- Mounts hard disk
- Creates user processes
- Sleeps until last user process died, ...

Daniel Gruss

- Kernel Thread
- Mounts hard disk
- Creates user processes
- Sleeps until last user process died, ...
- ... and unmounts the hard disk again
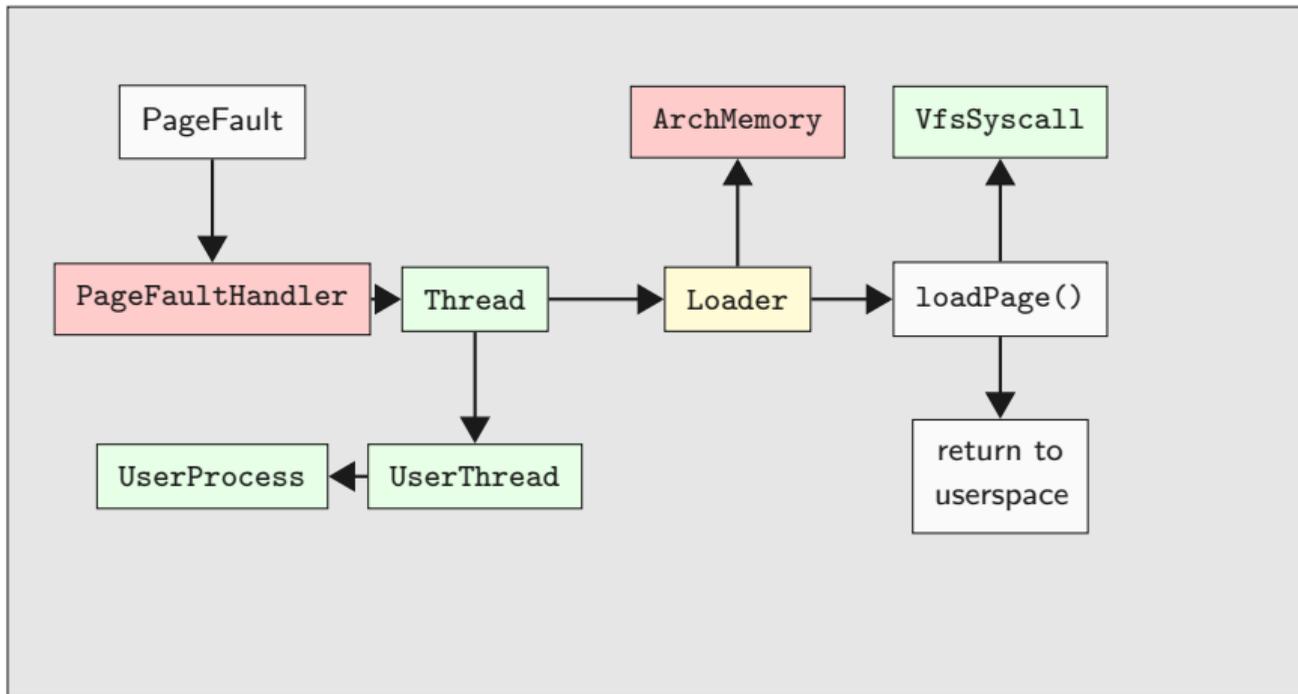
Status quo

- Derived from `Thread`

Status quo

- Derived from `Thread`
- Bad design is easier for you to improve ;)

Daniel Gruss

Status quo

- Derived from Thread
- Bad design is easier for you to improve ;)
- Executes binary code
- Has a virtual address space (Loader $\rightarrow$ ArchMemory)
- Has a userspace part and a kernel part

- Every 54.925439*ms*

- Implemented in `InterruptUtils.cpp`

```cpp
extern "C" void irqHandler_0()
{
  ArchCommon::drawHeartBeat();
  Scheduler::instance()->incTicks();
  Scheduler::instance()->schedule();
  ArchInterrupts::EndOfInterrupt(0);
  arch_contextSwitch();
}
```

Scheduler

- List of threads
- `schedule()` on IRQ0
- `schedule()` on IRQ65 (yield)
- `void Scheduler::addNewThread(Thread *thread);`
- Contains IdleThread (hlt if idle)
- Contains CleanupThread
    - Calls delete on dead threads

Scheduling a process

A process is scheduled:

- only if switch_to_userspace_ == 1
- Scheduler loads register values from ArchThreadRegisters member variable
- → Implicitly sets RIP from ArchThreadRegisters
- RIP initially points to binary entry point
- CPU switches to user mode and continues with given RIP

Daniel Gruss

userspace/libc/src/nonstd.c

```
extern int main ();

void _start ()
{
  exit ( main ());
}
```

libc is in userspace!

```
userspace/tests/helloworld.c
#include <stdio.h>
int main()
{
  puts("hello, world");
  return 0;
}
```

- Off-the-shelf Hello World Program
- Brian Kernigham, 1974

```
userspace/libc/printf.c
int puts(const char *output_string)
{
  // ...
    characters_written = write(STDOUT_FILENO, (void*) output_string,
   string_length);
  // ...
}
```

- In libc again
- Sanity checks
- Actually only a wrapper for syscall write

```
userspace/libc/write.c
ssize_t write(int file_descriptor, const void *buffer, size_t count)
{
  return __syscall(sc_write, file_descriptor,
          (long) buffer, count, 0x00, 0x00);
}
```

- POSIX!

- fwrite is a simple write wrapper

- Down the rabbit hole: __syscall

arch/x86/64/common/userspace/syscalls.c

```
size_t __syscall(size_t arg1, size_t arg2, size_t arg3,
                 size_t arg4, size_t arg5, size_t arg6)
{
  asm("int $0x80\n" : "=a"(arg1)
                    : "a"(arg1), "b"(arg2), "c"(arg3),
                      "d"(arg4), "S"(arg5), "D"(arg6));
  return arg1;
}
```

- Copy arguments to registers

- Issue interrupt 0x80

$\rightarrow$ switch to kernel space

```
arch/x86/64/common/source/arch_interrupts.S
arch_syscallHandler:
    pushall
    movq %rsp,%rdi
    movq $0,%rsi
    call arch_saveThreadRegisters
    call syscallHandler
```

```
InterruptUtils.cpp
extern "C" void syscallHandler() {
  thread->switch_to_userspace_ = 0;
  threadRegisters = thread->kernel_regs_;
  ArchInterrupts::enableInterrupts();

  auto ret = Syscall::syscallException(
                      thread->user_regs_->rax,
                      thread->user_regs_->rbx,
                      thread->user_regs_->rcx,
                      thread->user_regs_->rdx,
                      thread->user_regs_->rsi,
                      thread->user_regs_->rdi);
```

Syscall.cpp

```
size_t Syscall::syscallException(size_t syscall_number, size_t arg1,
    size_t arg2, size_t arg3, size_t arg4, size_t arg5) {
  switch (syscall_number)
  {
    // ...
    case sc_write:
      return_value = write(arg1, arg2, arg3);
      break;
    // ...
```

syscallException only calls the right methods with the right number of parameters

- `sc_write` constant defined in `Syscall.h`
- `Syscall::write` method is regular in-kernel C++

Daniel Gruss

```
size_t Syscall::write(size_t fd, pointer buffer, size_t size)
{
 if (fd == fd_stdout) //stdout
 {
  debug(SYSCALL, "Syscall::write: %.*s\n", (int)size, (char*) buffer);
  kprintf("%.*s", (int)size, (char*) buffer);
 }
 // ...
```

Daniel Gruss

```
common/include/console/debug.h
const size_t MAIN       = Ansi_Red     | ENABLED;
const size_t THREAD     = Ansi_Magenta | ENABLED;
const size_t USERPROCESS= Ansi_Cyan    | ENABLED;
const size_t PROCESS_REG= Ansi_Yellow  | ENABLED;
const size_t BACKTRACE  = Ansi_Red     | ENABLED;
const size_t USERTRACE  = Ansi_Red     | ENABLED;
//group memory management
const size_t PM         = Ansi_Green   | ENABLED;
const size_t KMM        = Ansi_Yellow;
```

Add your own debug tags!

"This is not the printf you are looking for"

- kprintf → SWEB Terminal
- kprintfd → Host
- Cannot be disabled by flags
- → Prefer debug

```c
#include <stdio.h>
int main()
{
  puts("hello, world");
  return 0;
}
```

What does return 0; do?

```
userspace/libc/src/nonstd.c
extern int main();

void _start()
{
  exit(main());
}
```

```
size_t Syscall::syscallException(...) {
    // ...
    case sc_exit:
      exit(arg1);
      break;
    // ...
}
void Syscall::exit(size_t exit_code) {
  debug(SYSCALL, "Syscall::EXIT: called, exit_code: %zd\n", exit_code)
    ;
  currentThread->kill();
}
```

```
void Thread::kill() {
  switch_to_userspace_ = 0;
  state_ = ToBeDestroyed;
  if (currentThread == this) {
    ArchInterrupts::enableInterrupts();
    Scheduler::instance()->yield();
  }
}
```

Recall: Scheduler will call delete on thread if state_ == ToBeDestroyed

```
UserProcess::~UserProcess() {
  delete loader_;
  vfs_syscall.close(fd_);
  delete working_dir_;
  process_registry_->processExit();
}
Thread::~Thread() {
  delete user_registers_;
  delete kernel_registers_;
}
```

THAT'S IT, I'VE HAD ENOUGH

Daniel Gruss

- Mouse driver

- Mouse driver
- Window manager

- Mouse driver
- Window manager
- Network driver

Daniel Gruss

- Mouse driver
- Window manager
- Network driver
- Soundblaster driver

- Mouse driver
- Window manager
- Network driver
- Soundblaster driver
- Gameboy emulator

Daniel Gruss

- Mouse driver
- Window manager
- Network driver
- Soundblaster driver
- Gameboy emulator
- 3D game engine

- Mouse driver
- Window manager
- Network driver
- Soundblaster driver
- Gameboy emulator
- 3D game engine
- Running it on a small ARM board with only 256KB RAM

Daniel Gruss

First steps

- Try out the tutorials on `http://iaik.tugraz.at/os`
  - Set up development environment
  - Implement your first syscall

Daniel Gruss

First steps

- Try out the tutorials on `http://iaik.tugraz.at/os`
  - Set up development environment
  - Implement your first syscall
- Get acquainted with the source code: Try out implementing things in SWEB

Daniel Gruss

First steps

- Try out the tutorials on `http://iaik.tugraz.at/os`
  - Set up development environment
  - Implement your first syscall
- Get acquainted with the source code: Try out implementing things in SWEB
- Start with the practicals NOW! (or rather already a week ago)

Daniel Gruss

- Base line SWEB: a user process is a (kernel) thread

Task 1: Multithreading

- Base line SWEB: a user process is a (kernel) thread
- We want: multiple threads per user process

Daniel Gruss

Task 1: Multithreading

- Base line SWEB: a user process is a (kernel) thread
- We want: multiple threads per user process
- What do we have to change?

Daniel Gruss

- Each thread has its own instances of some resources
  - id, stack, registers, status, ...

Task 1: Multithreading

- Each thread has its own instances of some resources
    - id, stack, registers, status, ...
- Other resources are shared among all threads
    - memory, files, ...

Daniel Gruss

- How to "use" multithreading?

Task 1: Multithreading

- How to "use" multithreading?
- Syscalls!
- Which ones?

Daniel Gruss

Task 1: Multithreading

- How to "use" multithreading?
- Syscalls!
- Which ones?
- You decide - but function names and arguments have to be POSIX-compatible!

Task 1: Multithreading

- How to "use" multithreading?
- Syscalls!
- Which ones?
- You decide - but function names and arguments have to be POSIX-compatible!
- Minimum requirements:

  pthread_create
  pthread_exit
  pthread_cancel
  pthread_join

Daniel Gruss

- By definition: Operating system is written by people who know what they do

- By definition: Operating system is written by people who know what they do
- User programs?

Syscalls (system calls)

- By definition: Operating system is written by people who know what they do

- User programs?

- System calls provide a "safe" interface

Daniel Gruss

Syscalls (system calls)

- By definition: Operating system is written by people who know what they do
- User programs?
- System calls provide a "safe" interface
- Control flow is transmitted to kernel code

Daniel Gruss

Syscalls (system calls)

- By definition: Operating system is written by people who know what they do
- User programs?
- System calls provide a "safe" interface
- Control flow is transmitted to kernel code
- Typical syscalls: `fork()`, `read()`, `write()`, `execve()`, `wait()`, `exit()`

Daniel Gruss

Syscalls (system calls)

- By definition: Operating system is written by people who know what they do
- User programs?
- System calls provide a "safe" interface
- Control flow is transmitted to kernel code
- Typical syscalls: `fork()`, `read()`, `write()`, `execve()`, `wait()`, `exit()`
- You will step through a syscall in the tutorial this week!

Daniel Gruss

- fork() creates a new process by duplicating the calling process
- The new process (=the child), is an exact duplicate of the calling process (=the parent)

- fork() creates a new process by duplicating the calling process
- The new process (=the child), is an exact duplicate of the calling process (=the parent)
- Interesting in combination with multithreading!

Daniel Gruss

- Replaces the current process image with a new process image

Additional Task: exec

- Replaces the current process image with a new process image
- exec with arguments $\rightarrow$ more points

Daniel Gruss

- Replaces the current process image with a new process image
- exec with arguments $\rightarrow$ more points
- fork()/exec() combination often used

- sleep() sets a thread asleep for a given number of seconds

- sleep() sets a thread asleep for a given number of seconds
- clock returns how much cpu time the current process consumed

Daniel Gruss

- exit() terminates the current process

Daniel Gruss

- `exit()` terminates the current process
- Already implemented, but ...

Additional Task: exit

- exit() terminates the current process
- Already implemented, but ...
- ... you will break the current implementation with multithreading

- I/O syscalls are already implemented, but …

- I/O syscalls are already implemented, but ...
- ... they use global (not process specific) file descriptors
- Why is that a problem?

- Threads need synchronization

Daniel Gruss

- Threads need synchronization
- Kernel has mutexes and condition variables

Daniel Gruss

Additional Task: Synchronization

- Threads need synchronization
- Kernel has mutexes and condition variables
- We want: mutexes, condition variables and semaphores, both in kernelspace and userspace

Daniel Gruss

Additional Task: Synchronization

- Threads need synchronization
- Kernel has mutexes and condition variables
- We want: mutexes, condition variables and semaphores, both in kernelspace and userspace
- Pure userspace implementation (except for initialization and for going to sleep)

Additional Task: Synchronization

- Threads need synchronization
- Kernel has mutexes and condition variables
- We want: mutexes, condition variables and semaphores, both in kernelspace and userspace
- Pure userspace implementation (except for initialization and for going to sleep)
- Implement test programs (Readers-Writers-Problem, Sleeping Barber, etc.)

Daniel Gruss

- Own ideas are the most fun!

Daniel Gruss

Additional Task: Your own ideas

- Own ideas are the most fun!
- See https://www.iaik.tugraz.at/teaching/materials/os/assignments/ for suggestions

Daniel Gruss

- Own ideas are the most fun!
- See https://www.iaik.tugraz.at/teaching/materials/os/assignments/ for suggestions
- Please note: Assignment 1 tasks will only be counted in Assignment 1 assessment

- Own ideas are the most fun!
- See `https://www.iaik.tugraz.at/teaching/materials/os/assignments/` for suggestions
- Please note: Assignment 1 tasks will only be counted in Assignment 1 assessment Assignment 2 tasks will only be counted in Assignment 2 assessment

Assignment 1

- Deadline Design-PoC: **See Timeline**
  - The design proof-of-concept comprises your implementation so far, as is. It will be graded by the test system and you will talk 30 minutes with your tutor about your progress.
- Deadline Implementation: **See Timeline**
- Minimum requirements:
  - At least **15** points from the two mandatory parts (multi threading and fork)
  - At least **1** point from the design proof-of-concept
  - At least **25** points in total
- 100% equals 50 points
- Tutors Points List A1 (not intended for student use)

In Assignment 1 you will acquire basic knowledge about the inner workings of multi threading and multi processing on modern operating systems.

Task 1: Multi Threading

You will start with the basic SWEB kernel which has neither multi threading nor multi processing implemented for userspace programs. Instead there is one thread per process. You have to figure out what to change to make SWEB support multi threading.

Threads share their address space with all other threads within the same process. However, each thread has its own stack, own set of cpu register values and maybe other resources. When the operating system interrupts a thread, it has to store these resources so that the thread execution can be continued later.

Interaction between userspace and kernelspace is only possible through syscalls. You can implement any syscall you like. However, for reasons of automated testing of your implementation you are required to **comply with the POSIX standard in terms of function names and parameters**.

You will need to implement at least the following syscalls:

1. `pthread_create` to create a new thread
2. `pthread_exit` to allow a thread to terminate itself
3. `pthread_cancel` to terminate another thread
4. `pthread_join` to wait for termination of another thread

Furthermore you are required to implement several user programs which demonstrate that your implementations works as expected.

Task 2: Fork

We want to give our user programs the ability to start new user programs. The POSIX compatible way of doing so, is to use the fork syscall (and the exec syscall). Fork creates an exact copy of the currently running process, with only the calling thread copied to the new process. The copied thread in the new process continues execution right after the fork call. We want to remind you to comply exactly to the POSIX standard for reasons of automated testing.

Additional Tasks

The two mandatory tasks are designed to give 30 points (of 50 points). We expect you to choose your own additional tasks. In the Tutors Points List A1 (not recommended for student use) you find a long list with numerous suggestions what you could implement. **DON'T** just use this list. Neither you nor we will be happy with the result. These are bonus tasks, come up with your own ideas, come up with your own improved variants of features.

We strongly encourage implementing your own interesting ideas, as long as they are related to this course, by giving you points. But, before you start, ask your tutor. For tasks that fall into one of the topics of Assignment 2 you can't get any points in Assignment 1 (and the other way round).

Tutors Point List

- Don't use it.
- No explanation. No details on how much you have to do for which point.
- Not suitable to choose tasks.
- Not suitable to estimate your points (points vary a lot!).

Daniel Gruss

Tutors Point List, `pthread_cancel` example ∎

- Tutors Point List says **2 points**
- "ah, actually cancel only works up to 5 times, then it stops working" → 0-1 points
- "actually, we have implemented it including cancelstate, canceltype, cleanup push, pthread keys, correct cancel and join interaction" without any problems → maybe 8-12 points
- → **You see?** The **2 points** doesn't tell you much! Can be a lot more, can be a lot less, all depends on how much you do.
- 2 points refers to a minimal correct working function that implements the most basic functionality of that syscall function

Daniel Gruss

Mandatory vs. Bonus

- "Is it mandatory to implement every sentence and everything that is mentioned on the (for example) `pthread_cancel` man page?"
- No. Minimal correctly working cancel is 2 points. Everything beyond (cancel types, cancel states, etc.) is all elective/bonus.
- → **70%** of the points you will collect in the OS exercises will be elective/bonus.

Daniel Gruss

Get those tutorials done asap and start coding!