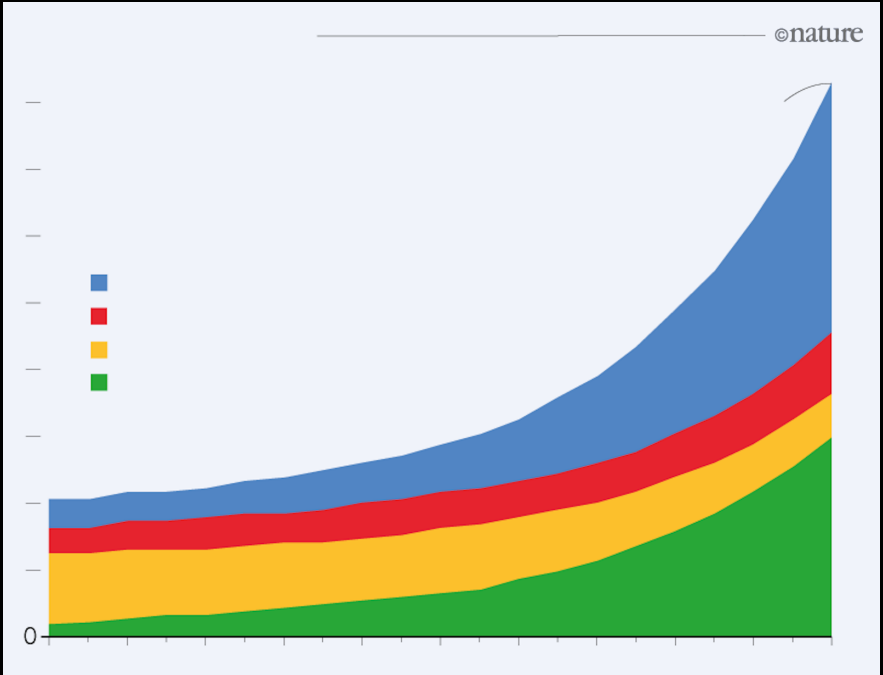
VU Operating Systems

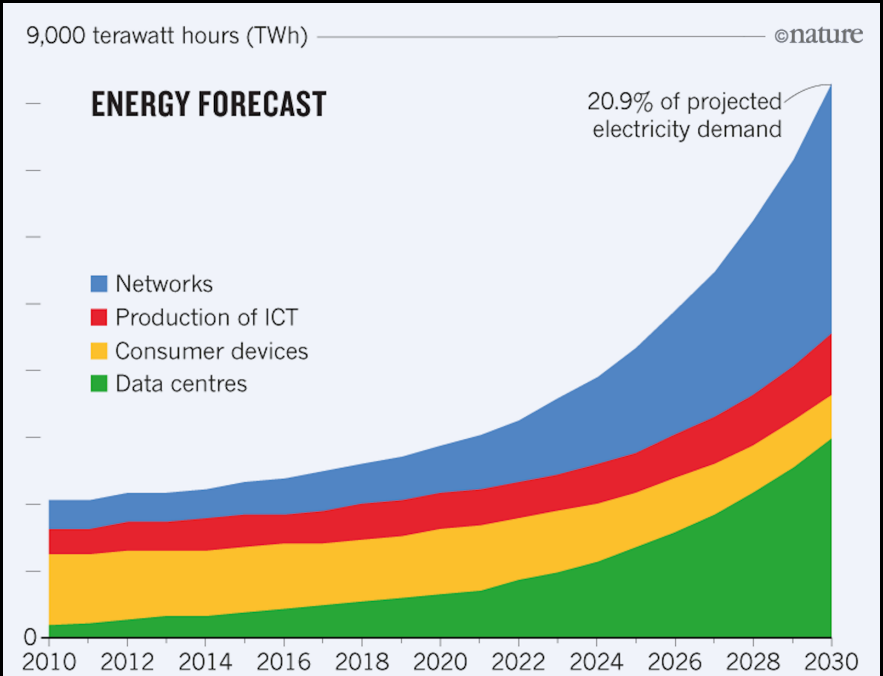
Daniel Gruss

2022-03-04

IAIK – Graz University of Technology

Why?


©nature
0


9,000 terawatt hours (TWh)
©nature
ENERGY FORECAST
20.9% of projected electricity demand
Networks
Production of ICT
Consumer devices
Data centres
0
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

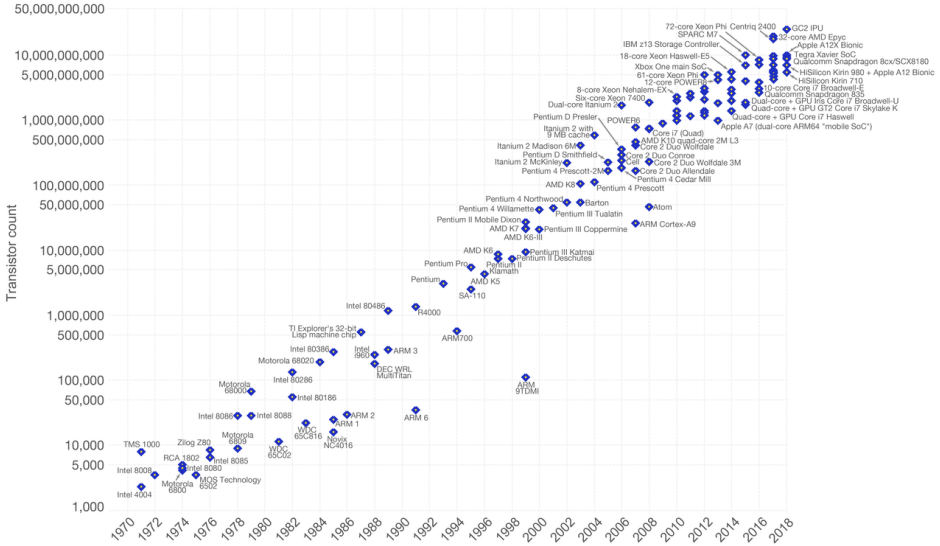
half a page table per process + 1 for the kernel

0.35%

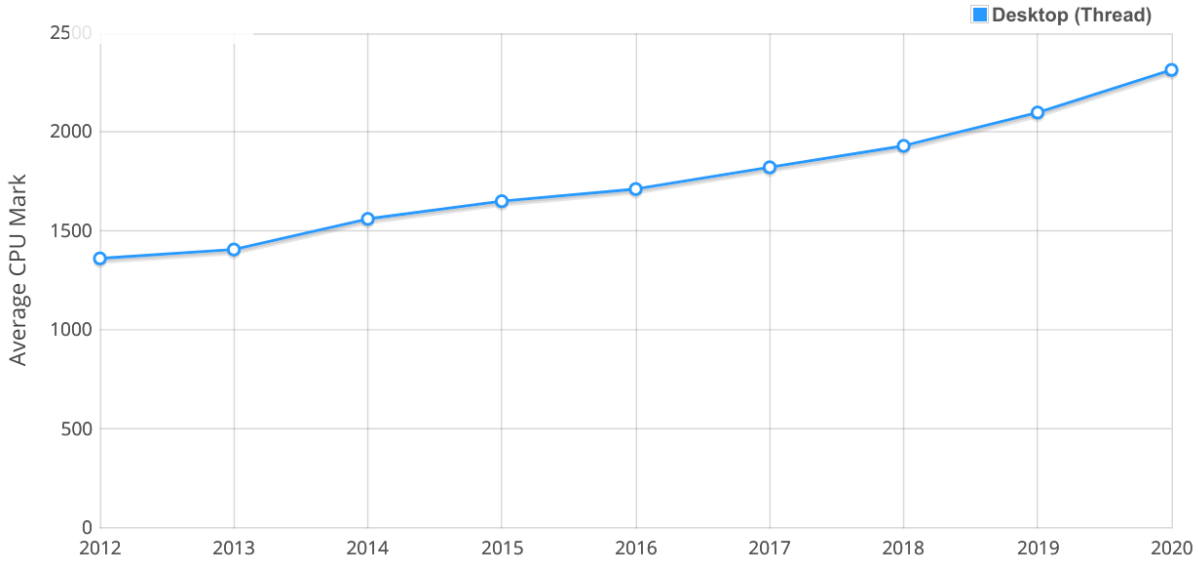
Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Our World in Data

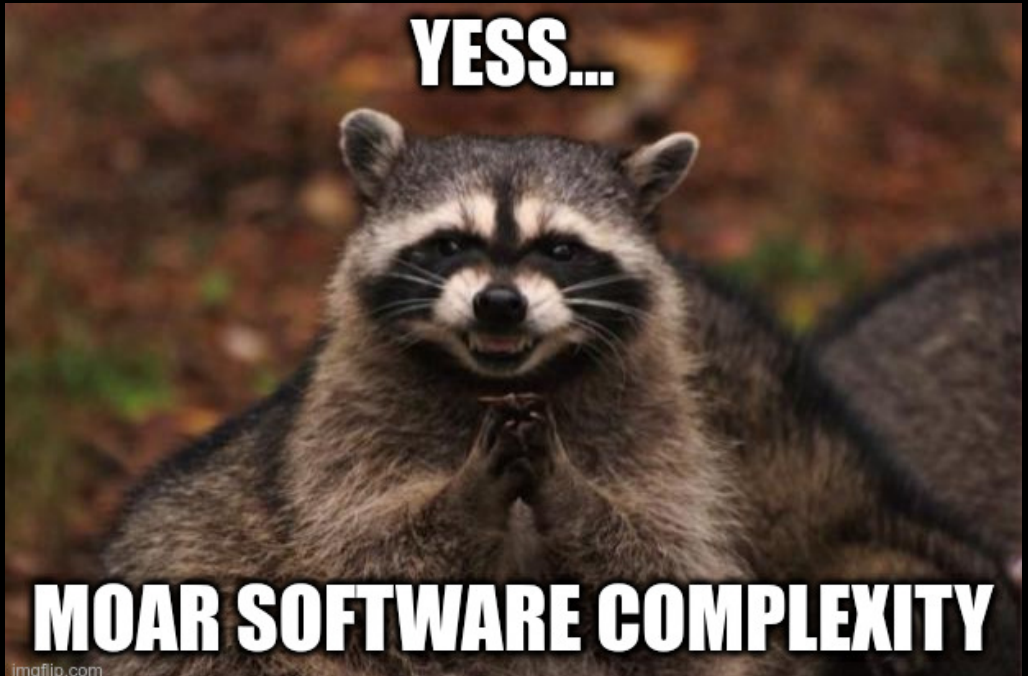
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.

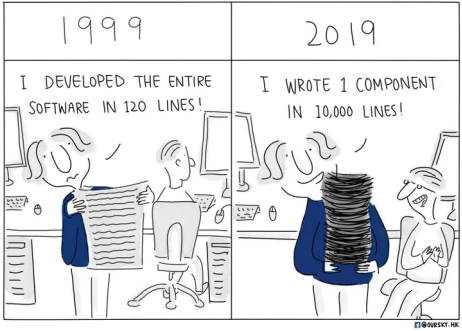


Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.


Desktop (Thread)
Average CPU Mark
2500
2000
1500
1000
500
0
2012
2013
2014
2015
2016
2017
2018
2019
2020


THE END IS NEAR
OF MOORE'S LAW


YESS...
MOAR SOFTWARE COMPLEXITY
imgflip.com

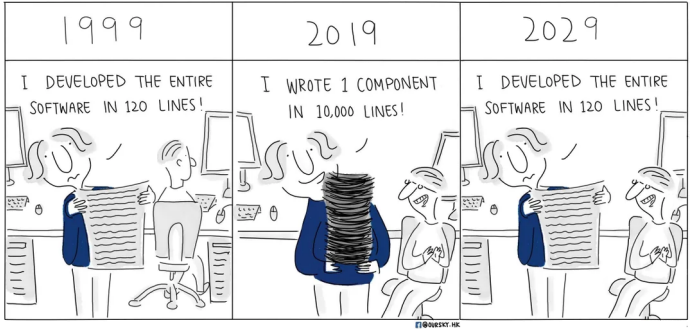

1999
I DEVELOPED THE ENTIRE SOFTWARE IN 120 LINES!
2019
I WROTE 1 COMPONENT IN 10,000 LINES!
@OURSKY.HK

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
(v) Lisp	2.27	(c) Pascal	3.02	(v) 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	(v) 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


1999
I DEVELOPED THE ENTIRE SOFTWARE IN 120 LINES!
2019
I WROTE 1 COMPONENT IN 10,000 LINES!
2029
I DEVELOPED THE ENTIRE SOFTWARE IN 120 LINES!
@OURSKY.HK

What we expect from you

- Knowledge from earlier lectures
  - ESP, OOP1(!), SLP(!!), CON, ...

What we expect from you

- Knowledge from earlier lectures
  - ESP, OOP1(!), SLP(!!), CON, ...
- Reasonable C/C++ experience
- Team work + Time management

What we expect from you

Check whether you are prepared:

- "If I would do ESP/OOP1/SLP/CON again I would **be able** to get a 1 on my own, without any help, without any problems, and without investing much time."
- Self assess for which of those 4 courses this statement fits you.
- Hint: ask a colleague whether he agrees with your self-assessment ;)

What we expect from you

- 4 out of 4: Congratulations, you are well prepared!

What we expect from you

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared $\rightarrow$ additional time investment necessary.

What we expect from you

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared $\rightarrow$ additional time investment necessary.
- 2 out of 4: You are not prepared $\rightarrow$ huge time investment or positive grade at risk.

What we expect from you

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared $\rightarrow$ additional time investment necessary.
- 2 out of 4: You are not prepared $\rightarrow$ huge time investment or positive grade at risk.
- 1 out of 4: You are seriously unprepared $\rightarrow$ even with huge time investment positive grade is at risk.

What we expect from you

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared $\rightarrow$ additional time investment necessary.
- 2 out of 4: You are not prepared $\rightarrow$ huge time investment or positive grade at risk.
- 1 out of 4: You are seriously unprepared $\rightarrow$ even with huge time investment positive grade is at risk.
- 0 out of 4: ...

How much effort is it?

- Depends significantly on your knowledge and experience ...

How much effort is it?

- Depends significantly on your knowledge and experience ...
- ... and on your team members.

How much effort is it?

- 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

How much effort is it?

- 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

“I am afraid now!”

- No need to be afraid!

“I am afraid now!”

- No need to be afraid!
- Before COVID, 35%-45% of students got a 1 in OS, every semester.

“I am afraid now!”

- No need to be afraid!
- Before COVID, 35%-45% of students got a 1 in OS, every semester.
- Is it easy?

“I am afraid now!”

- No need to be afraid!
- Before COVID, 35%-45% of students got a 1 in OS, every semester.
- Is it easy?
- No.

“I am afraid now!”

- No need to be afraid!
- Before COVID, 35%-45% of students got a 1 in OS, every semester.
- Is it easy?
- No.
- Is it doable?

“I am afraid now!”

- No need to be afraid!
- Before COVID, 35%-45% of students got a 1 in OS, every semester.
- Is it easy?
- No.
- Is it doable?
- Yes, very much so.

“I’m still afraid!”

- No one should be afraid of OS.

“I’m still afraid!”

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

“I’m still 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

“I’m still 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

“I’m still 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?

“I’m still 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?

“I’m still 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!

From Duke university

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.

From Duke university

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.

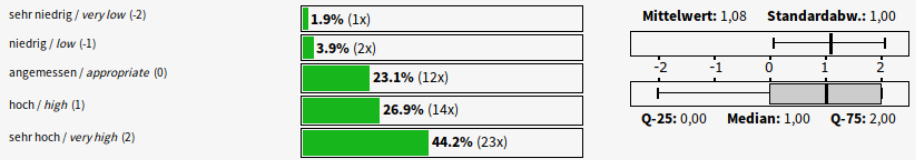
From Duke university

Note to students

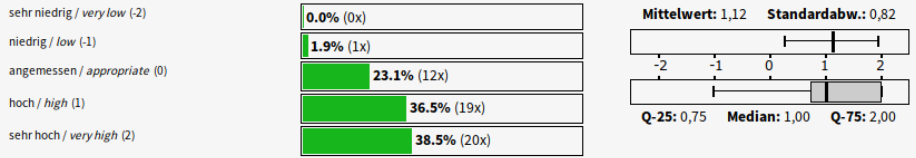
While we have made an effort to simplify these projects for you, most Duke students find these projects sufficiently difficult to dominate their lives during the one-semester course. A common misconception from earlier semesters is that we are sadistic individuals who enjoy seeing students suffer. Actually,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

From our evaluations

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? (52 x beantwortet)

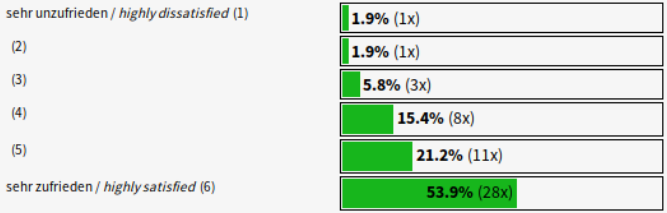


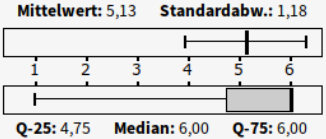
Wie beurteilen Sie die inhaltlichen Anforderungen (Schwierigkeit) dieser LV? /
How would you rate the content requirements (difficulty) of this course? (52 x beantwortet)



From our evaluations

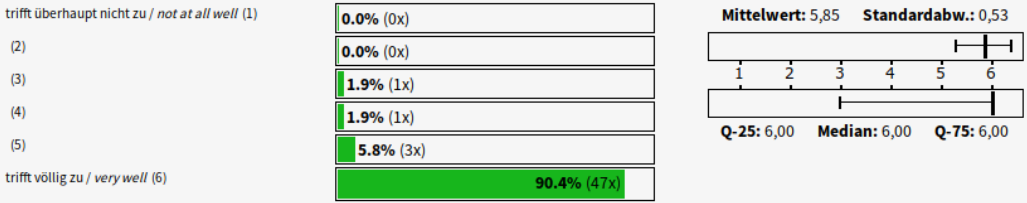
Wie zufrieden sind Sie mit der LV insgesamt? /
How satisfied are you with this course in general? (52 x beantwortet)





From our evaluations

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)." (52 x beantwortet)



uSTL hilfreich?

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

uSTL hilfreich?

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

uSTL hilfreich?

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.

uSTL hilfreich?

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.

uSTL hilfreich?

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.

uSTL hilfreich?

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.

Iteratoren verlieren die Gültigkeit, wo sie es nicht sollten

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

Effizienz?

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

From this course's evaluations

Was gefällt Ihnen an dieser LV besonders gut? 28 Antwort(en) vorhanden.

- Der "Klick" Moment wenn man endlich verstanden hat worums geht und die Einstellung zur LV von "Sehr abgeneigt" zu "Begeistert" umschwingt. [WS16/17]

Haben die vielen Assertions geholfen? i

- Assertions sind eine Wunderwaffe beim Debuggen ;)
- wenn man selber welche einfügt, dann sind sie noch hilfreicher ;)
- gegen ende hin waren ca 1/3 der zeilen Asserts - Hat jedoch dem tutor nicht so gut gefallen weil dadurch der code unleserlich war

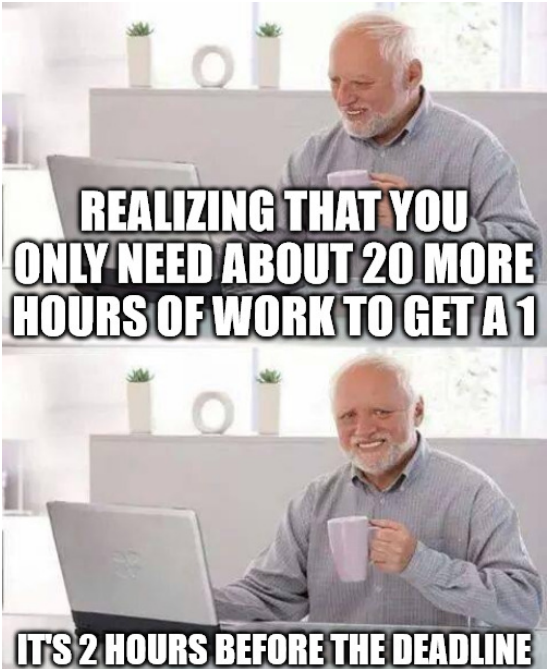
Feedback

- 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."

Typical problems

- 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++


REALIZING THAT YOU ONLY NEED ABOUT 20 MORE HOURS OF WORK TO GET A 1
IT'S 2 HOURS BEFORE THE DEADLINE

Teaching assistants

- Will help you with all your problems
  - Especially: if you're stuck on a problem for more than a few hours $\rightarrow$ ask
- Interactive Programming units
- Student Debates
- Design reviews
- Question hours
- Review meetings (Abgabegespräche)

Channels

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

Channels

- Website: `https://iaik.tugraz.at/os`
- Discord
- studo
- Email: os@iaik.tugraz.at
- Consultation hour: send us an email
- *During the question days: less activity on all channels ;)*

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

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

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.

Git is mandatory

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

Git is mandatory (2)

- Personal maximum number of points

Git is mandatory (2)

- 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

Git is mandatory (2)

- 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

Gitlab

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

Gitlab

- 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

Participation

- Every team member has to participate:

Participation

- Every team member has to participate:
  - Making coffee,

Participation

- Every team member has to participate:
  - Making coffee,fetching pizza,

Participation

- Every team member has to participate:
  - Making coffee,fetching pizza,etc.

Participation

- Every team member has to participate:
  - Making coffee,fetching pizza,etc. is not enough

Participation

- Every team member has to participate:
  - Making coffee,fetching pizza,etc. is not enough
- We expect all members to have a high level overview of design and implementation

Participation

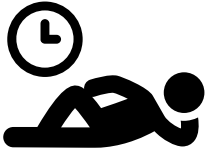
- Every team member has to participate:
  - Making coffee,fetching pizza,etc. is not enough
- We expect all members to have a high level overview of design and implementation
- We expect every member to be able to read, explain and change their own implementation, even if it's the code of another team member

Participation

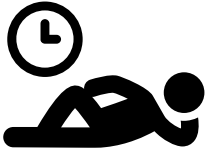
- Every team member has to participate:
  - Making coffee,fetching pizza,etc. is not enough
- We expect all members to have a high level overview of design and implementation
- We expect every member to be able to read, explain and change their own implementation, even if it's the code of another team member
- Otherwise: 0 points

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

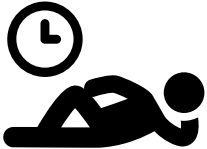
What if you as a team don't fit together?



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

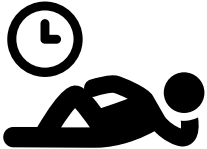


What if you as a team don't fit together?



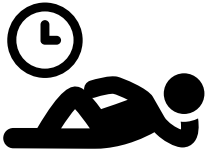
- Set internal deadlines for your team members

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



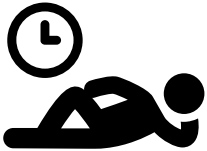
- Set internal deadlines for your team members
  - No idea how to split the work and choose good internal deadlines? Ask tutor for recommendations!

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



- 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

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



- 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

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

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

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

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

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

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

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

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

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

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)

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)

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?

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? → **Talk to me**, we'll figure out a solution

POSIX

- Existing syscall wrapper functions are POSIX-compatible

POSIX

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

POSIX

- 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

Testing

- Goal: a stable and fault tolerant operating system
- How?

Testing

- Goal: a stable and fault tolerant operating system
- How? By writing test programs

Testing

- Goal: a stable and fault tolerant operating system
- How? By writing test programs
- Think of test cases while designing and implementing
- Think of basic test scenarios as well as corner cases

Testing

- 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

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

Testing

- Bad: Not knowing of a problem until the review meeting

Testing

- Bad: Not knowing of a problem until the review meeting
- Better: Knowing of a problem but not solving it (probably because time ran out)

Testing

- Bad: Not knowing of a problem until the review meeting
- Better: Knowing of a problem but not solving it (probably because time ran out)
- Best: Knowing of a problem sufficiently before the deadline and solving it (maybe with the help of a teaching assistant)

Submissions

- 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`

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: Peter Lipp <peter.lipp@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!

Assessment A1

Reference (=100%)

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

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!

Assessment A2

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!

Assessment of the practicals

Minimum requirements

- A1: 25 of 50 points
- A2: 25 of 50 points

Assessment of the practicals

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%

Assessment

To pass the class, you have to acquire

- overall at least 55%

Assessment

To pass the class, you have to acquire

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

Assessment

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

Assessment: Second Chance

- Didn't pass any assignment, exam, design debate?

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

Assessment: Second Chance

- Didn't pass any assignment, exam, design debate?
- That's bad, but we have a second chance for everyone.

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

Assessment: Second Chance

- Didn't pass any assignment, exam, design debate?
- That's bad, but we have a second chance for everyone.
- DD $\rightarrow$ Second Chance DD with Daniel

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

Assessment: Second Chance

- Didn't pass any assignment, exam, design debate?
- That's bad, but we have a second chance for everyone.
- DD $\rightarrow$ Second Chance DD with Daniel
- PoC $\rightarrow$ Second Chance PoC with Daniel

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

Assessment: Second Chance

- Didn't pass any assignment, exam, design debate?
- That's bad, but we have a second chance for everyone.
- DD $\rightarrow$ Second Chance DD with Daniel
- PoC $\rightarrow$ Second Chance PoC with Daniel
- Exercise $\rightarrow$ Second Chance Exercise with Daniel and Tutor

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

Assessment: Second Chance

- Didn't pass any assignment, exam, design debate?
- That's bad, but we have a second chance for everyone.
- DD $\rightarrow$ Second Chance DD with Daniel
- PoC $\rightarrow$ Second Chance PoC with Daniel
- Exercise $\rightarrow$ Second Chance Exercise with Daniel and Tutor
- Exam $\rightarrow$ Second Chance Exam with Daniel

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

Assessment: Getting a grade

Marks

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

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

Plagiarism

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

Plagiarism

- 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 $\rightarrow$ teams are questioned
- Both teams: 0 points in either case
- At least one team: "Ungültig/Täuschung" (with all its consequences)

Plagiarism

- Do not provide your source code to other teams

Plagiarism

- 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

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

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 ;)

Questions so far?

History and the others

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

History and the others

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

History and the others

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

History and the others

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

History of SWEB

- BS KU 2004/2005

History of SWEB

- BS KU 2004/2005
- Advanced group of students together with Philip Lawatsch and Bernhard Tittelbach

History of SWEB

- BS KU 2004/2005
- Advanced group of students together with Philip Lawatsch and Bernhard Tittelbach
- Many subsequent projects

History of SWEB

- BS KU 2004/2005
- Advanced group of students together with Philip Lawatsch and Bernhard Tittelbach
- Many subsequent projects
- BS KU: since 2007 SWEB only

Base Line SWEB

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

Base Line SWEB

- 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

What is possible in SWEB

- Mouse driver

What is possible in SWEB

- Mouse driver
- Window manager

What is possible in SWEB

- Mouse driver
- Window manager
- Network driver

What is possible in SWEB

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

What is possible in SWEB

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

What is possible in SWEB

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

What is possible in SWEB

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

First steps

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

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

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)

Task 1: Multithreading

- 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

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?

Task 1: Multithreading

- 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, ...

Task 1: Multithreading

- How to "use" multithreading?

Task 1: Multithreading

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

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`

Syscalls (system calls)

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

Syscalls (system calls)

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

Syscalls (system calls)

- By definiton: Operating system is written by people who know what they do
- User programs?
- System calls provide a “safe” interface

Syscalls (system calls)

- By definiton: 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

Syscalls (system calls)

- By definiton: 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()`

Syscalls (system calls)

- By definiton: 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!

Task 2: fork

- `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)

Task 2: fork

- `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!

Additional Task: exec

- 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

Additional Task: exec

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

Additional Task: sleep/clock

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

Additional Task: sleep/clock

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

Additional Task: exit

- `exit()` terminates the current process

Additional Task: exit

- `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

Additional Task: I/O syscalls

- I/O syscalls are already implemented, but ...

Additional Task: I/O syscalls

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

Additional Task: Synchronization

- Threads need synchronization

Additional Task: Synchronization

- Threads need synchronization
- Kernel has mutexes and condition variables

Additional Task: Synchronization

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

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.)

Additional Task: Your own ideas

- Own ideas are the most fun!

Additional Task: Your own ideas

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

Additional Task: Your own ideas

- 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

Additional Task: Your own ideas

- 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

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!).

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.

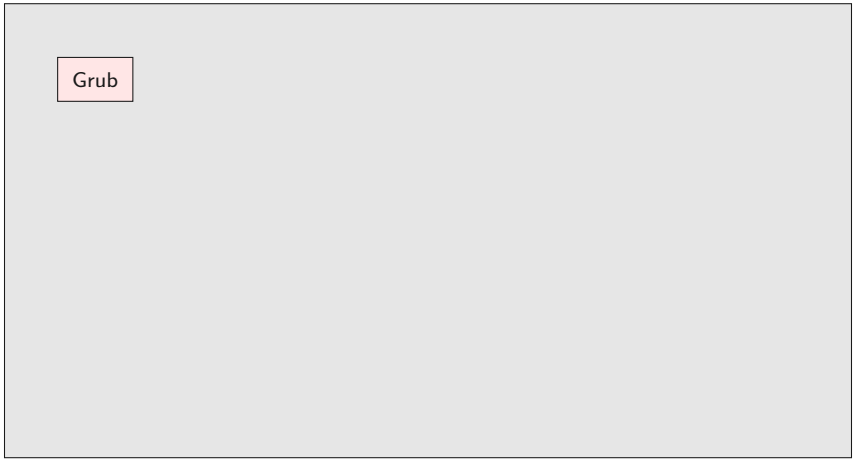
- 2 points refers to a minimal correct working function that implements the most basic functionality of that syscall function

Mandatory vs. Bonus

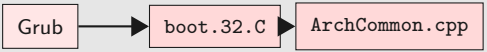
- "Is it mandatory to implement every sentence and everything that is mentioned on the (for example) `pthread_cancel` man page?"
- No. Minimal correctly working cancel is 2 points. Everything beyond (cancel types, cancel states, etc.) is all elective/bonus.

→ **70%** of the points you will collect in the OS exercises, will be elective/bonus.

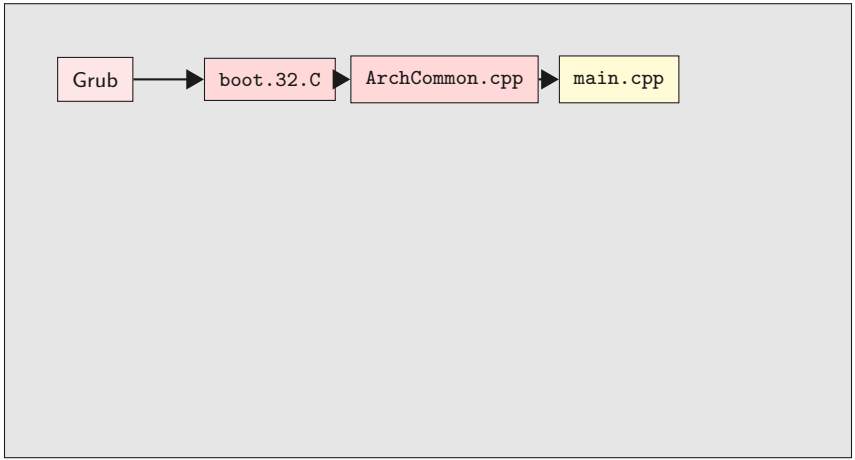
Overview



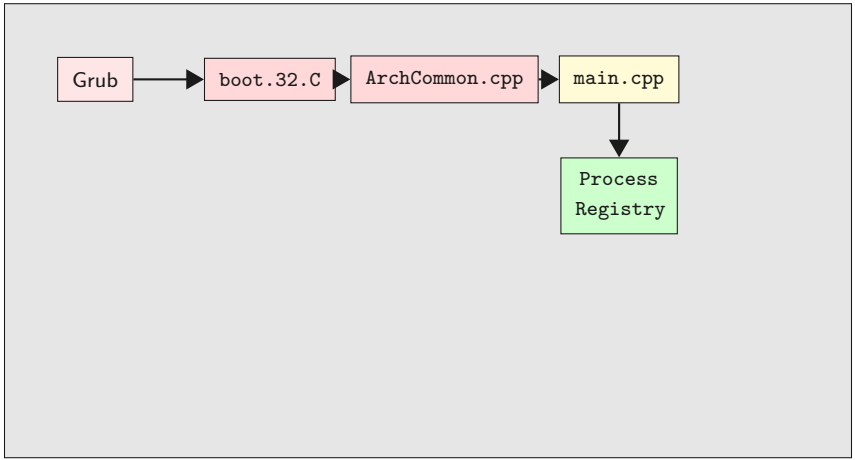
Overview



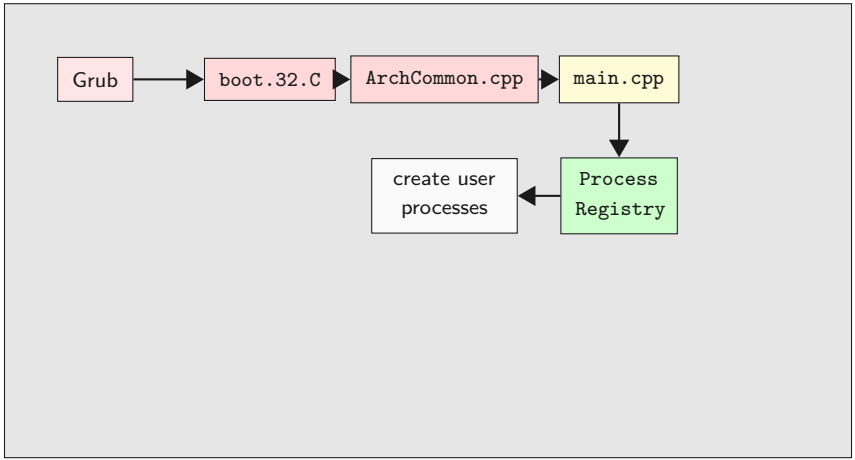
Overview



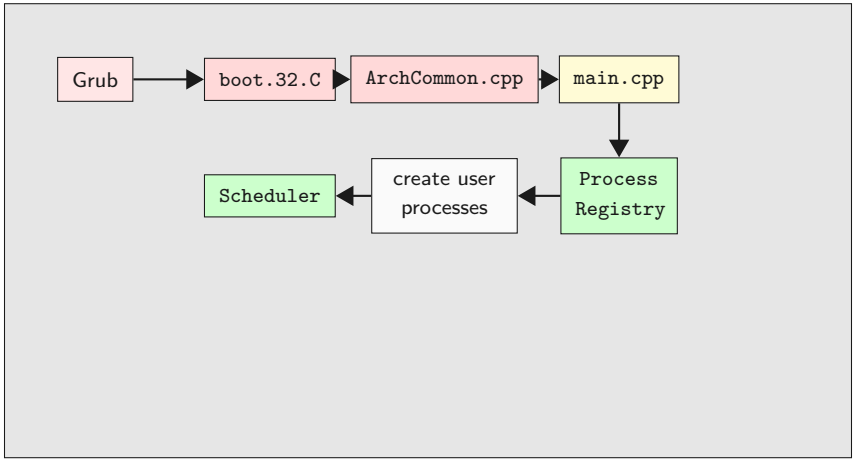
Overview



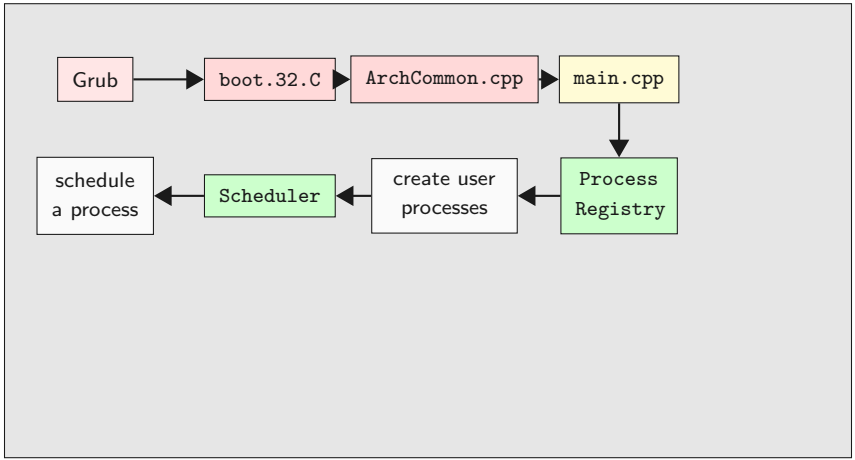
Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes

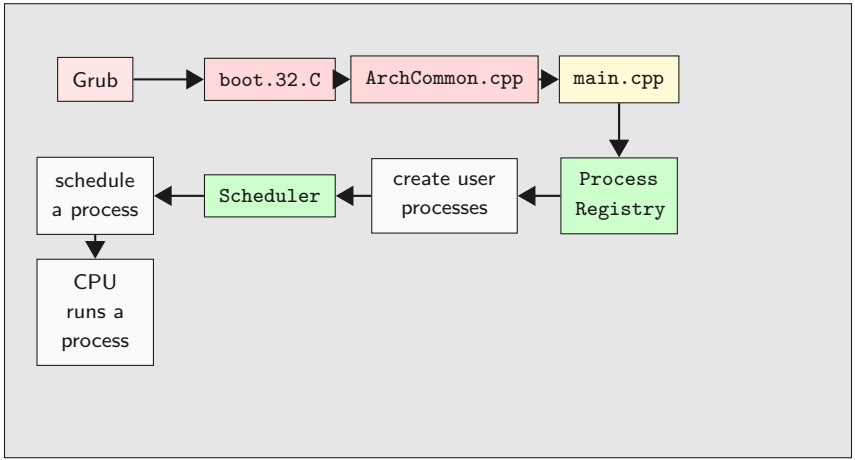
Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler

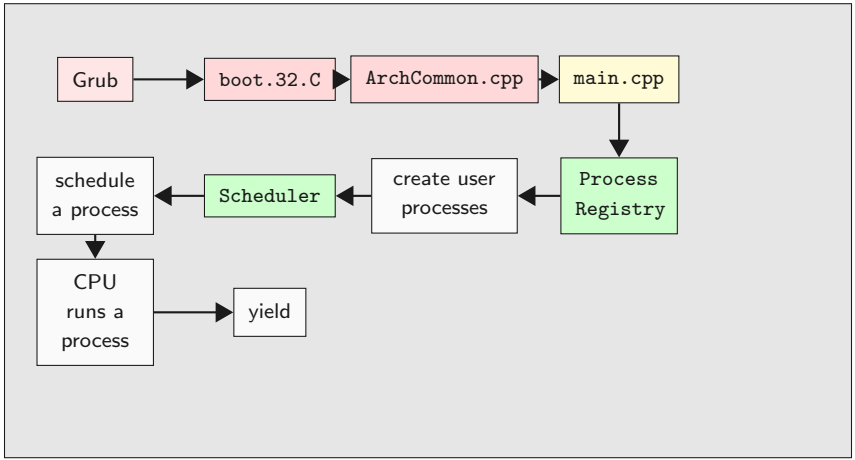
Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process

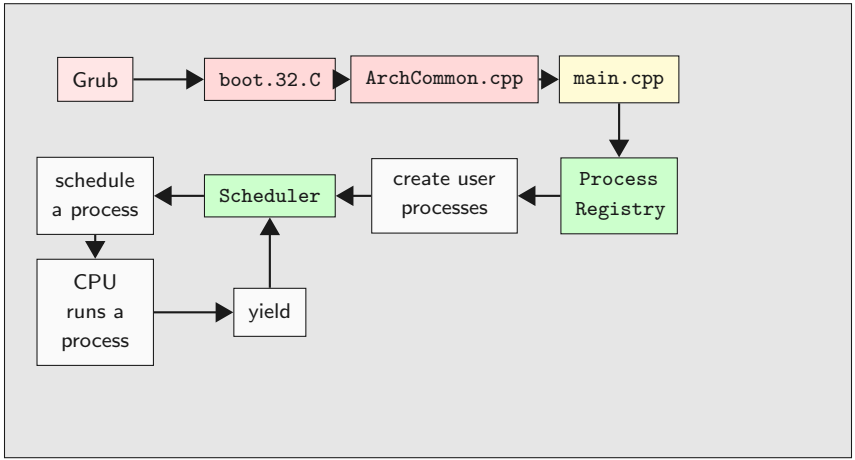
Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process
CPU runs a process

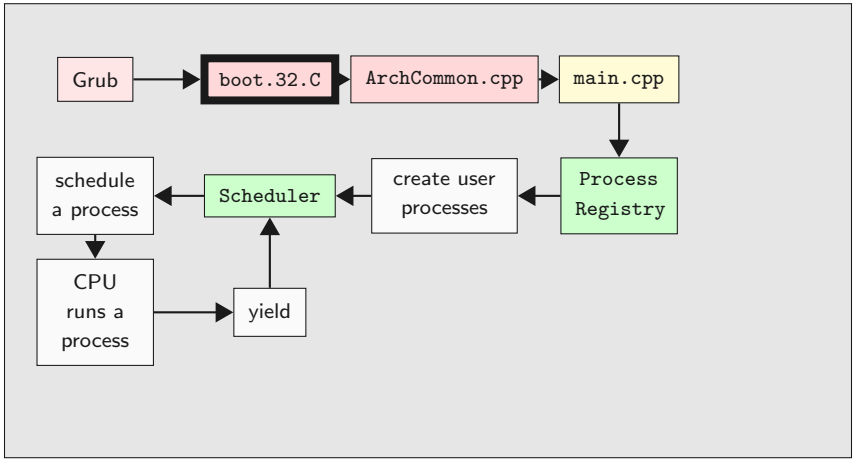
Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process
CPU runs a process
yield

Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process
CPU runs a process
yield

Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process
CPU runs a process
yield

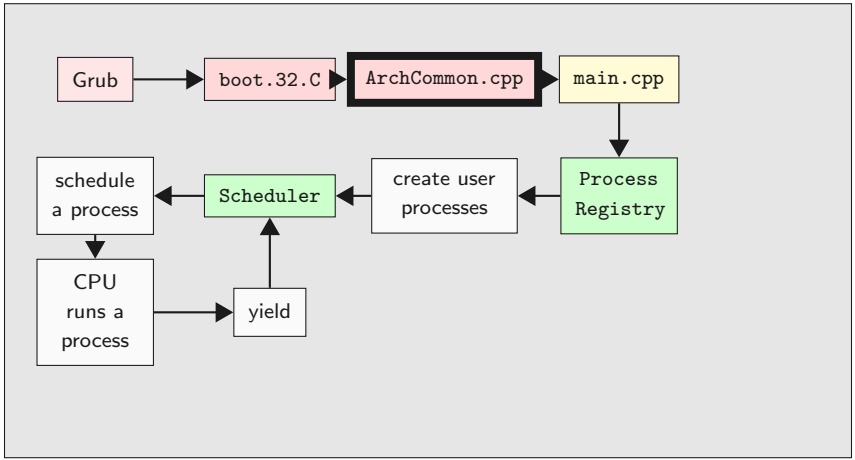
boot.32.C

```
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));
```

boot.32.C

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

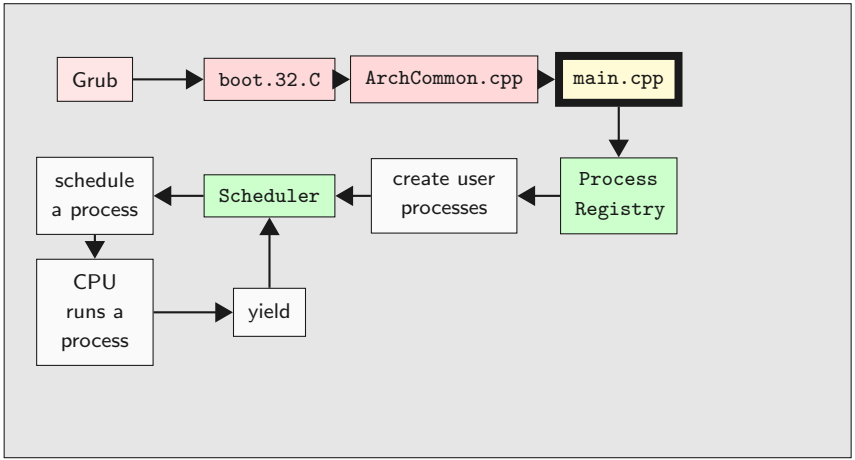
Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process
CPU runs a process
yield

ArchCommon.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));
```

Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process
CPU runs a process
yield

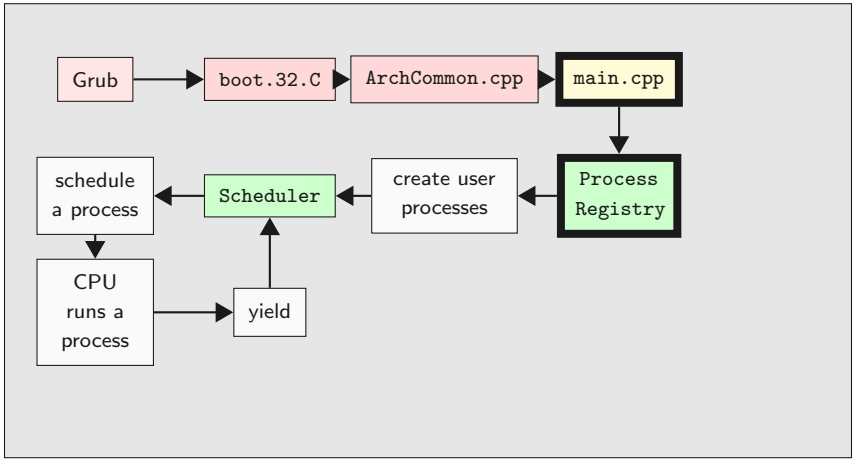
main.cpp

```
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
```

`main.cpp`

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

Overview


Grub
boot.32.C
ArchCommon.cpp
main.cpp
Process Registry
create user processes
Scheduler
schedule a process
CPU runs a process
yield

ProcessRegistry

```
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]);
  }
```

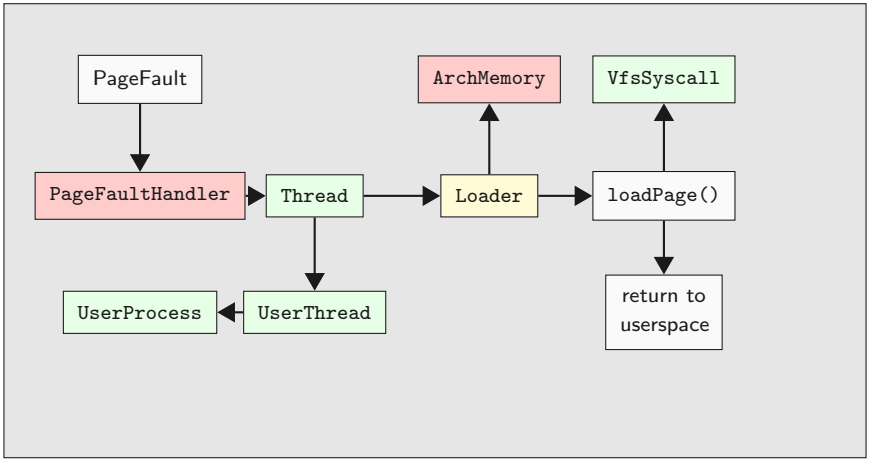
ProcessRegistry

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

ProcessRegistry

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

Loading User Processes



`UserProcess`

Status quo

- Derived from `Thread`

`UserProcess`

Status quo

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

UserProcess

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

Timer Interrupt

- Every 54.925439ms
- Implemented in `InterruptUtils.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

Entry point

userspace/libc/src/nonstd.c

```
extern int main();

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

libc is in userspace!

Hello World!

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

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

puts?

```
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`

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, 0, 0);
}
```

- POSIX!
- `fwrite` is a simple `write` wrapper
- Down the rabbit hole: `__syscall`

__syscall

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

```
void __syscall()
{
  asm("int $0x80");
}
```

- Function call copies arguments to registers
- Issue interrupt 0x80

→ switch to kernel space

Syscall Dispatching in Kernel

arch/x86/64/common/source/arch_interrupts.S

```
arch_syscallHandler:
    pushall
    movq %rsp,%rdi
    movq $0,%rsi
    call arch_saveThreadRegisters
    call syscallHandler
```

Syscall dispatching (higher level)

InterruptUtils.cpp

```
extern "C" void syscallHandler() {
  thread->switch_to_userspace_ = 0;
  threadRegisters = thread->kernel_regs_;
  ArchInterrupts::enableInterrupts();
  thread->user_regs_->rax =
    Syscall::syscallException(
                  thread->user_regs_->rdi,
                  thread->user_regs_->rsi,
                  thread->user_regs_->rdx,
                  thread->user_regs_->rcx,
                  thread->user_regs_->r8,
                  thread->user_regs_->r9);
```

Syscall handling

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

Syscall handling

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

Syscall::write

```
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);
 }
 // ...
```

Debug Flags

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`

Process termination

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

What does `return 0;` do?

Process termination (2)

userspace/libc/src/nonstd.c

```
extern int main();

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


AANND
IT'S GONE

Exit Syscall

```
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();
}
```

Thread termination

```
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`

Thread deletion

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

Final Advise

Get in touch with the source code!