

VU Operating Systems

Daniel Gruss

2020-03-02

IAIK – Graz University of Technology

Organizational Details

SWEB

Assignment 1

Booting SWEB

Organizational Details

SWEB

Assignment 1

Booting SWEB



Peter Lipp



Daniel Gruss



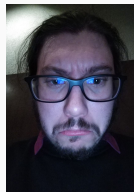
Michael Schwarz



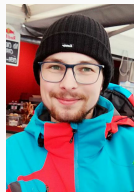
Florian Kargl



Luca Mayr



Christian Rieger



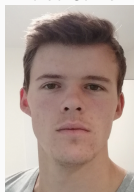
David Schögler



Stefan Steininger



Hanna Müller



Alexander Ertl



Martin Glasner



Lukas Maar



Fabian Rauscher



Weisstainer

- Will help you with all your problems
- Interactive Programming units
- Design discussions
- Design reviews
- Question hours
- Review meetings (Abgabegespräche)

- Wiki: <https://teaching.iaik.tugraz.at/bs>

- Wiki: <https://teaching.iaik.tugraz.at/bs>
- Newsgroup
 - Organizational: tu-graz.lv.bs
 - SWEB specific: tu-graz.lv.bs.sweb

- Wiki: `https://teaching.iaik.tugraz.at/bs`
- Newsgroup
 - Organizational: `tu-graz.lv.bs`
 - SWEB specific: `tu-graz.lv.bs.sweb`
- IRC: `irc://irc.freenode.net/sweb`

- Wiki: <https://teaching.iaik.tugraz.at/bs>
- Newsgroup
 - Organizational: tu-graz.lv.bs
 - SWEB specific: tu-graz.lv.bs.sweb
- IRC: <irc://irc.freenode.net/sweb>
- Email: bs-helpline@iaik.tugraz.at
- Consultation hour: send us an email
- *During the question days: less activity on all channels ;)*

02.03.-06.03. Interactive Programming

02.03.-06.03. Interactive Programming

09.03.-13.03. Interactive Virtual Memory Tutorial

09.03. Assignment A1 Tutorial

02.03.-06.03. Interactive Programming

09.03.-13.03. Interactive Virtual Memory Tutorial

09.03. Assignment A1 Tutorial

16.03.-20.03. Design Discussions A1

27.03. 18:00 Deadline Design-PoC A1

- 02.03.-06.03.** Interactive Programming
- 09.03.-13.03.** Interactive Virtual Memory Tutorial
- 09.03.** Assignment A1 Tutorial
- 16.03.-20.03.** Design Discussions A1
- 27.03.** 18:00 Deadline Design-PoC A1
- 30.03.-03.04.** Review Design-PoC A1
- 24.04.** Question Day A1
- 26.04.-30.04.** Question Days A1
- 30.04.** 18:00 Deadline Implementation A1 - **Thursday!**
- 04.05.-08.05.** Review Meetings A1

05.05. Assignment A2 Tutorial

11.05.-15.05. Design Discussions A2

22.05. 18:00 Deadline Design-PoC A2

05.05. Assignment A2 Tutorial

11.05.-15.05. Design Discussions A2

22.05. 18:00 Deadline Design-PoC A2

25.05.-29.05. Review Design-PoC A2

15.06.-19.06. Question Days A2

19.06. 18:00 Deadline Implementation A2 - **Friday!**

22.06.-26.06. Review Meetings A2

- Knowledge from earlier lectures
 - Rechnerorganisation(!), SNP (!!), ...

- Knowledge from earlier lectures
 - Rechnerorganisation(!), SNP(!!), ...
- Reasonable C/C++ experience
- Team work + Time management

Check whether you are prepared:

- “If I would do ESP/SEP/RO/SNP again I would 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 ;)

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

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared, but you can catch up with some additional time investment.

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared, but you can catch up with some additional time investment.
- 2 out of 4: You are not prepared for the course. It will be a huge time investment to get any positive grade at all.

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared, but you can catch up with some additional time investment.
- 2 out of 4: You are not prepared for the course. It will be a huge time investment to get any positive grade at all.
- 1 out of 4: You are seriously unprepared. Even with a huge time investment it's very unsure whether you can make it.

- 4 out of 4: Congratulations, you are well prepared!
- 3 out of 4: You are not well prepared, but you can catch up with some additional time investment.
- 2 out of 4: You are not prepared for the course. It will be a huge time investment to get any positive grade at all.
- 1 out of 4: You are seriously unprepared. Even with a huge time investment it's very unsure whether you can make it.
- 0 out of 4: Don't waste your time. Consider doing this course at a later point in your studies when you are prepared and have the required knowledge to start with this course.

- Depends significantly on your knowledge and experience ...

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

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

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.

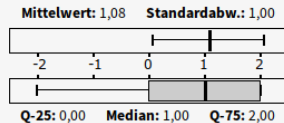
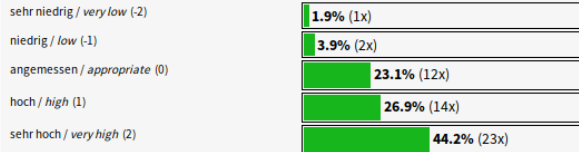
Note to students

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.

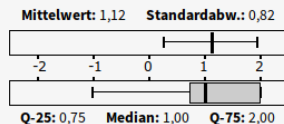
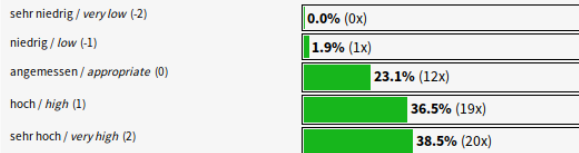
Note to students

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? (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)



Wie zufrieden sind Sie mit der LV insgesamt? /

How satisfied are you with this course in general? (52 x beantwortet)

sehr unzufrieden / highly dissatisfied (1)

1.9% (1x)

(2)

1.9% (1x)

(3)

5.8% (3x)

(4)

15.4% (8x)

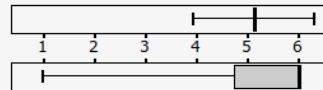
(5)

21.2% (11x)

sehr zufrieden / highly satisfied (6)

53.9% (28x)

Mittelwert: 5,13 Standardabw.: 1,18



Q-25: 4,75 Median: 6,00 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)." (52 x beantwortet)

trifft überhaupt nicht zu / *not at all well* (1)

0.0% (0x)

(2)

0.0% (0x)

(3)

1.9% (1x)

(4)

1.9% (1x)

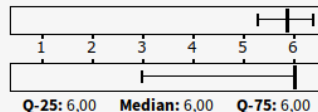
(5)

5.8% (3x)

trifft völlig zu / *very well* (6)

90.4% (47x)

Mittelwert: 5,85 Standardabw.: 0,53



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

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

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

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]

- 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

- 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.”

- Bad time management
- Problems with working in a team
- No C/C++ experience
- Not willing to learn and use C/C++

- 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

- 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

- Only code! No design document!
- Proof of concept implementation
- You get up to 5 points, depending on the test system score of your submission.

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

- 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

- 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 has to participate:

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

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

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

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

- 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

- 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

- 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

- 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

- 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

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

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

- 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

- Writing numerous test programs is unavoidable

- 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

- 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

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

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

- 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)

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

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

- 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 → small deduction
- Do this at the same time:
 - Mail to helpline!
 - Submit and retag your fixed version!

Reference (=100%)

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

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 in the Wiki!

Reference (=100%)

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

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 in the Wiki!

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

- Discussions with other teams are appreciated
- But: **no collaboration!**
- We check for plagiarism
- Similarities → teams are questioned

- 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)

- 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

- 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 → 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 → plagiarism
 - Your own code (not exactly the same team)
→ not allowed

- 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 → plagiarism
 - Your own code (not exactly the same team)
→ not allowed
 - Your own code (exactly the same team)
→ allowed, probably not the best idea ;)

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
- Otherwise you will get less points

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

- two written exams: 20% each
- the practicals: 60%

To pass the class, you have to acquire

- overall at least 55%

To pass the class, you have to acquire

- overall at least 55%
- at least 50% of the possible points in one written exam and at least 33% in the other

To pass the class, you have to acquire

- overall at least 55%
- at least 50% of the possible points in one written exam and at least 33% in the other
- at least 50% of the possible points in the practicals

Marks

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

Questions so far?

Organizational Details

SWEB

Assignment 1

Booting SWEB

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

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

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

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

- BS KU 2004/2005

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

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

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

- 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

- Mouse driver

- Mouse driver
- Window manager

- Mouse driver
- Window manager
- Network driver

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

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

- 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

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

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

- Try out the tutorials on `http://teaching.iaik.tugraz.at/bs`
 - 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)

Organizational Details

SWEB

Assignment 1

Booting SWEB

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

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

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

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

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

- How to “use” multithreading?

- How to “use” multithreading?
- Syscalls!
- Which ones?

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

- 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`

- By definiton: 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?

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

- 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

- 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()`

- 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!

- `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!

- Replaces the current process image with a new process image

- Replaces the current process image with a new process image
- exec with arguments → more points

- Replaces the current process image with a new process image
- `exec` with arguments → 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

- `exit()` terminates the current process

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

- `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

- Threads need synchronization
- Kernel has mutexes and condition variables

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

- 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)

- 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.)

- Own ideas are the most fun!

- Own ideas are the most fun!
- See <https://teaching.iaik.tugraz.at/bs/assignments> for suggestions

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

- Own ideas are the most fun!
- See <https://teaching.iaik.tugraz.at/bs/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

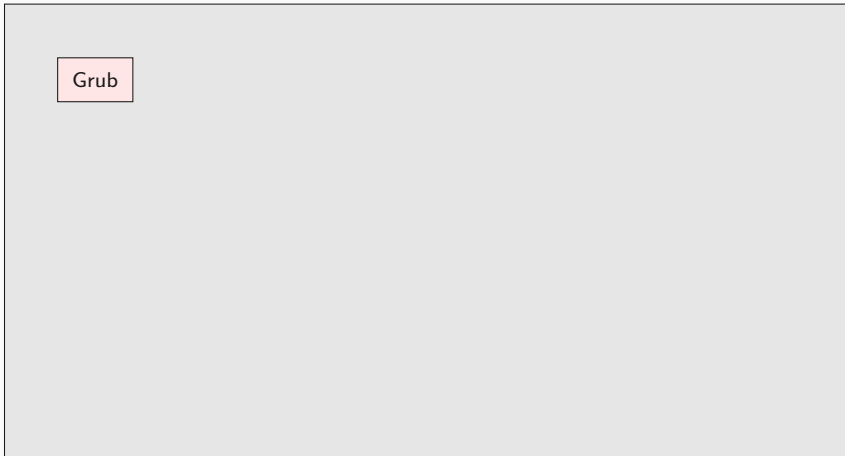
- We use this for grading and we are legally expected to put this online.
- No explanation. No details on how much you have to do for which point.
- Not suitable to choose tasks.
- Not suitable to estimate your points (points vary a lot!).

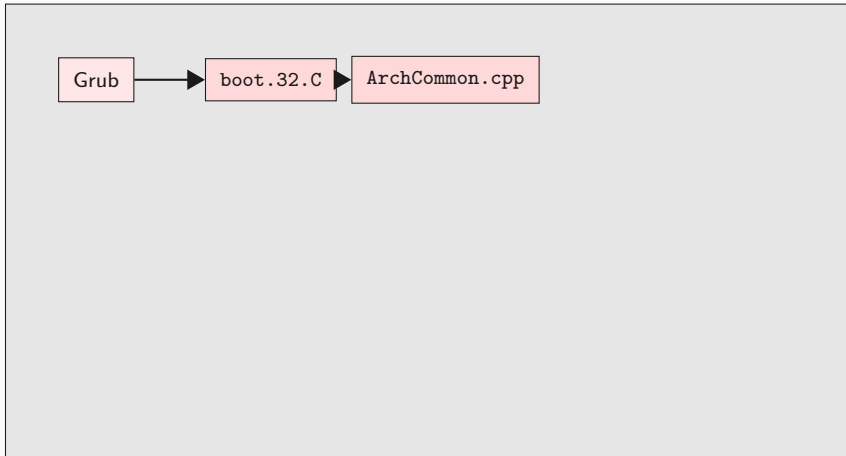
Organizational Details

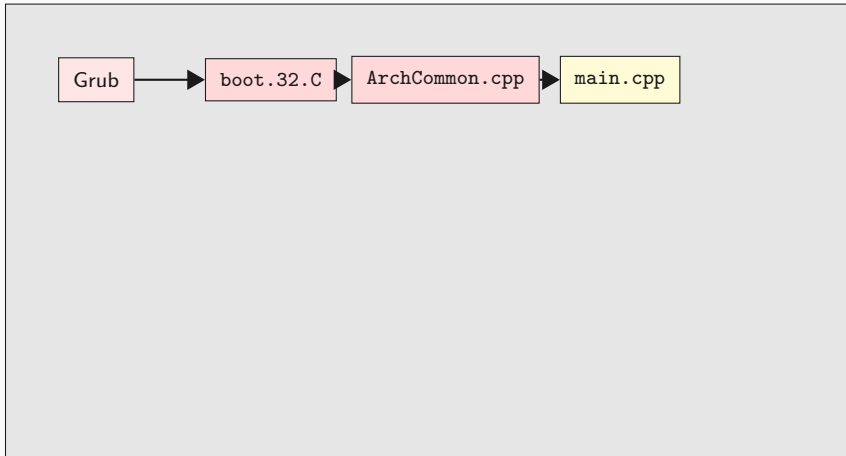
SWEB

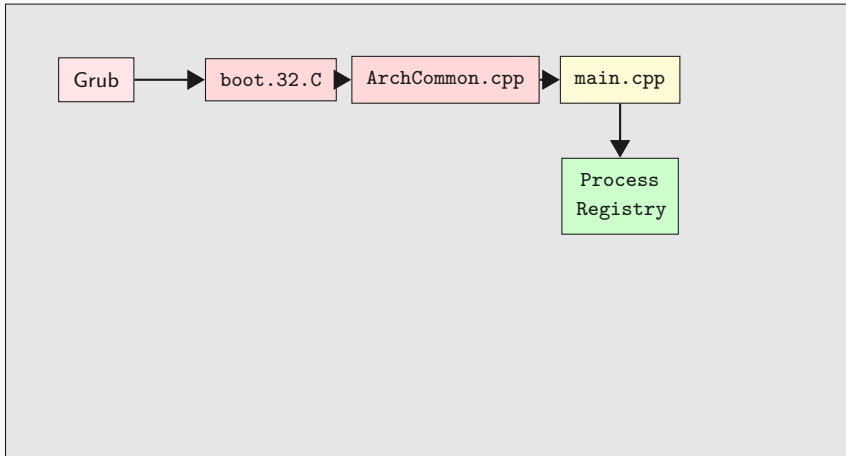
Assignment 1

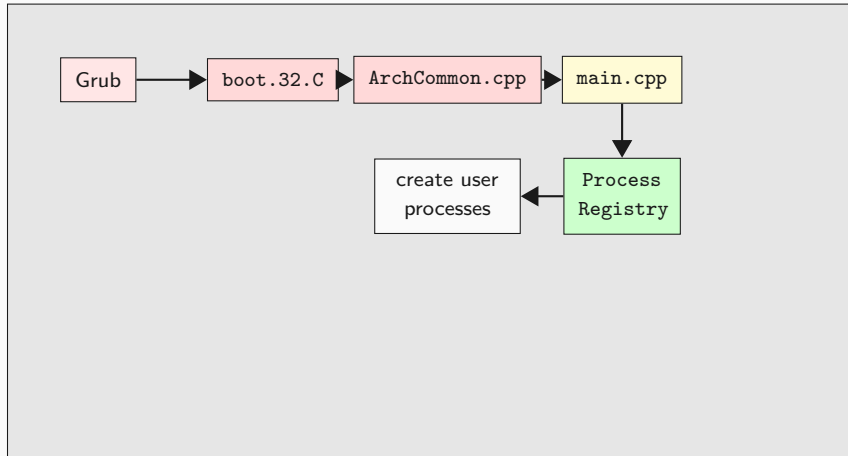
Booting SWEB

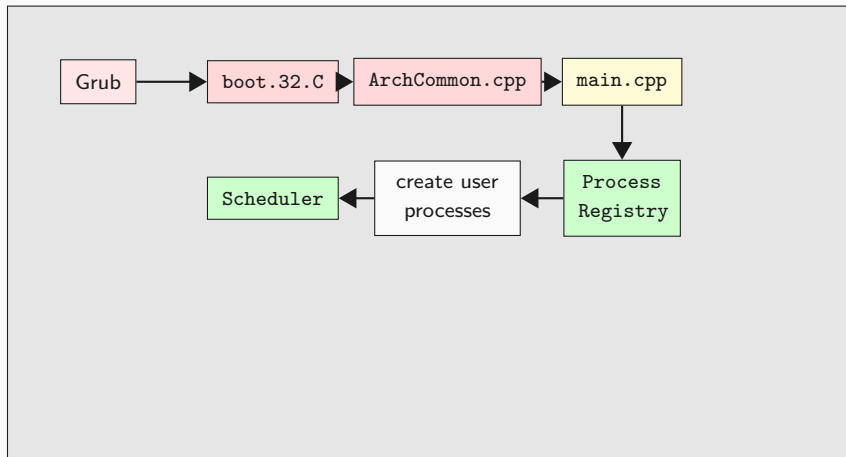


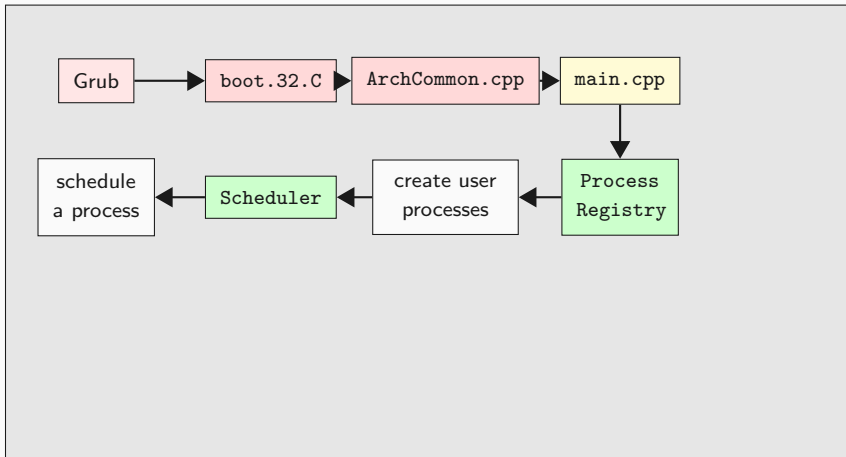


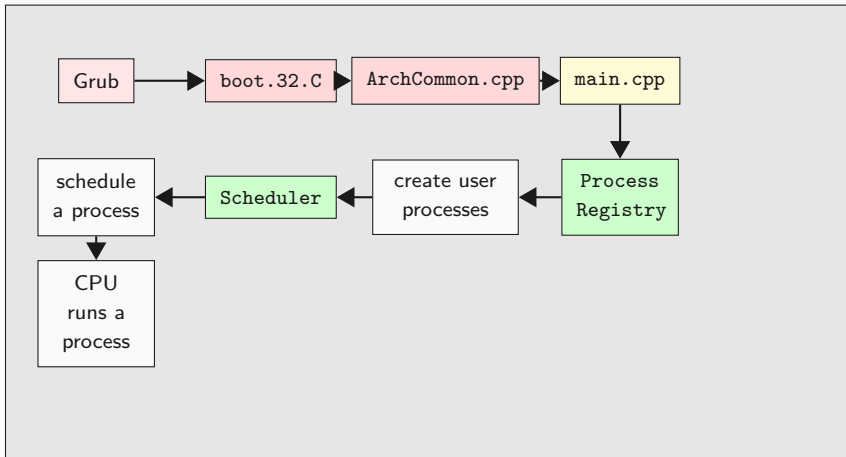


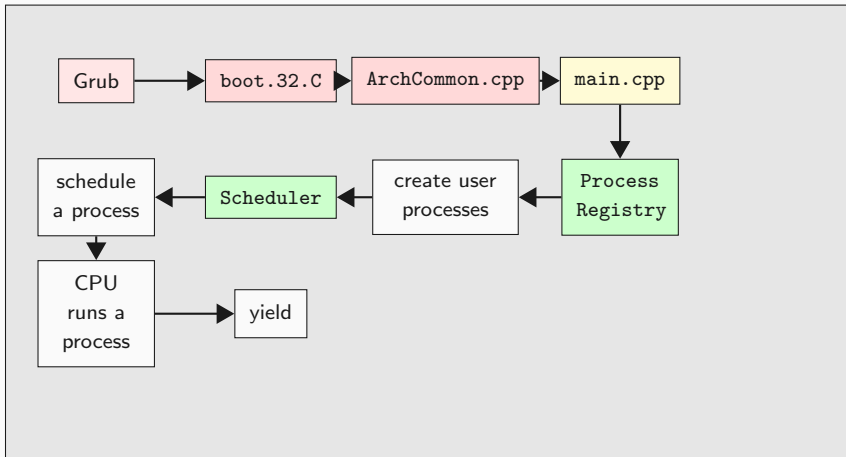


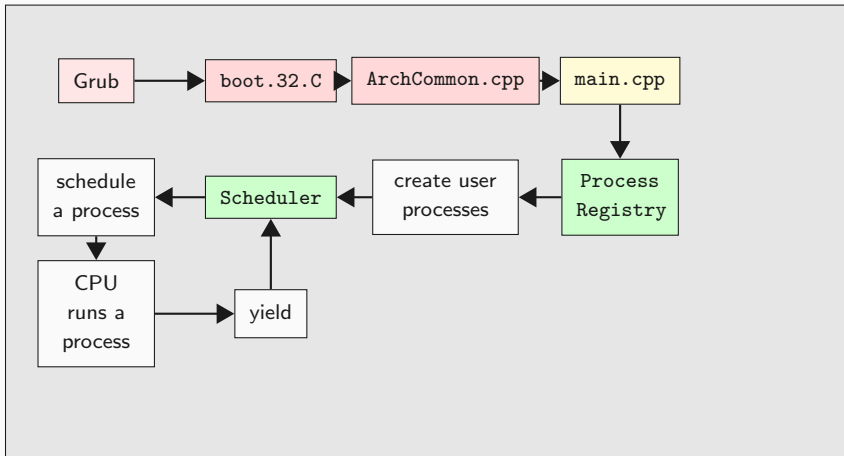


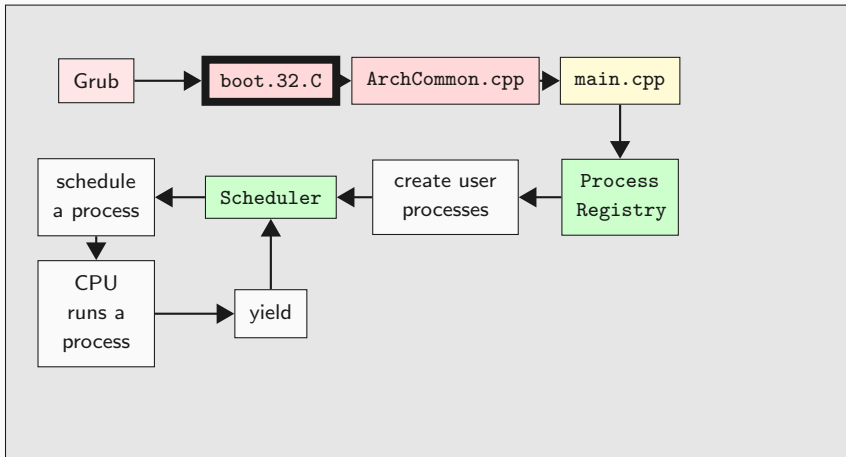






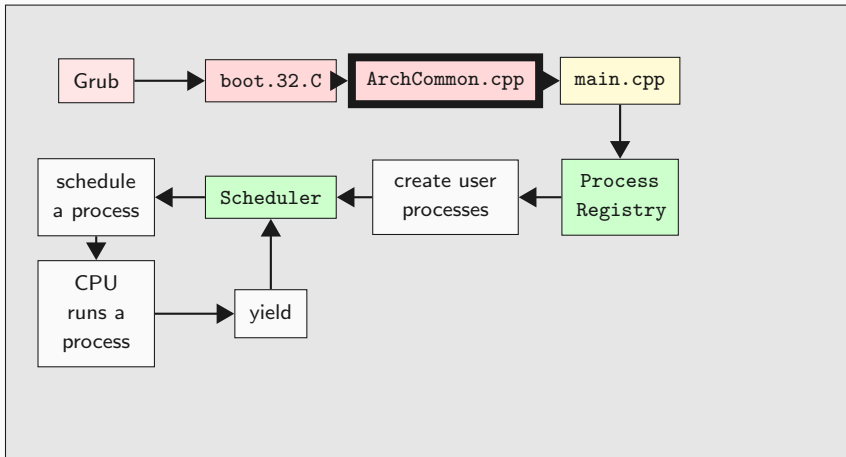




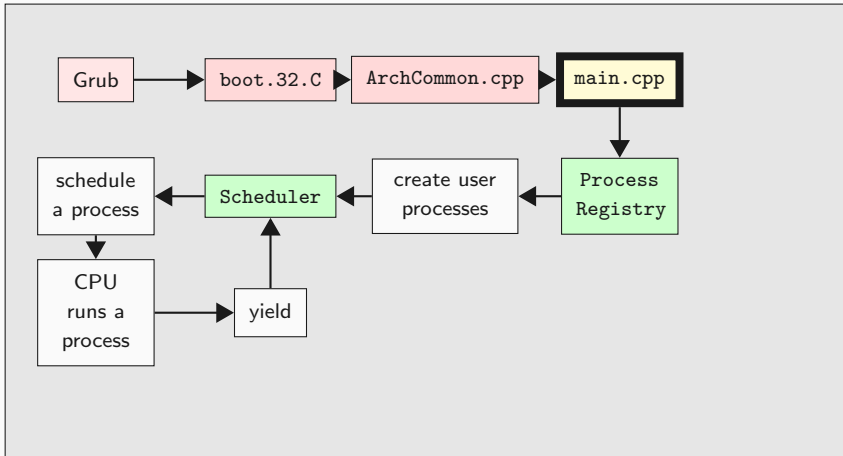


```
extern "C" void entry()
{
    PRINT("Booting...\n");
    // ...
    PRINT("Initialize Kernel Paging Structs\n");
    // ...
    PRINT("Enable Paging...\n");
    // ...
    PRINT("Calling entry64()...\n");
    asm("ljmp %[cs], $entry64-BASE\n" : : [cs] "i"(KERNEL_CS));
}
```

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

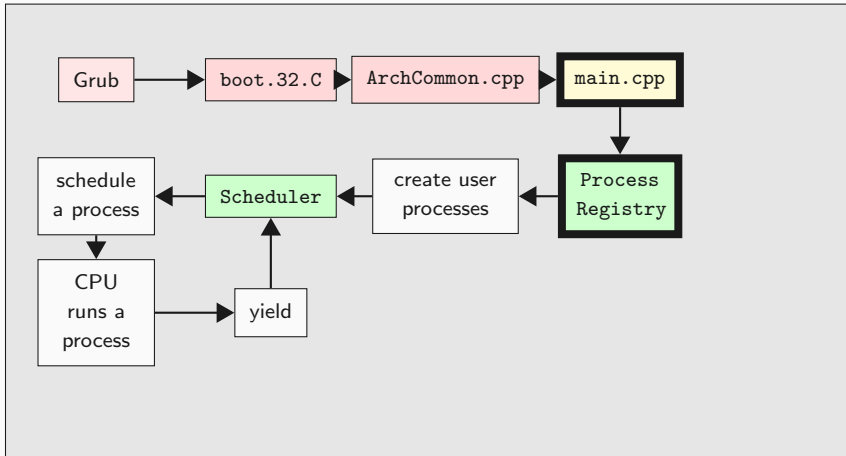


```
extern "C" void entry64()
{
    // ...
    PRINT("Switch to our own stack...\n");
    asm("mov %[stack], %%rsp\n"
        "mov %[stack], %%rbp\n" : : [stack]"i"(boot_stack + 0x4000));
    PRINT("Loading Long Mode Segments...\n");
    // ...
    PRINT("Calling startup()...\n");
    asm("jmp *%[startup]" : : [startup]"r"(startup));
}
```



```
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

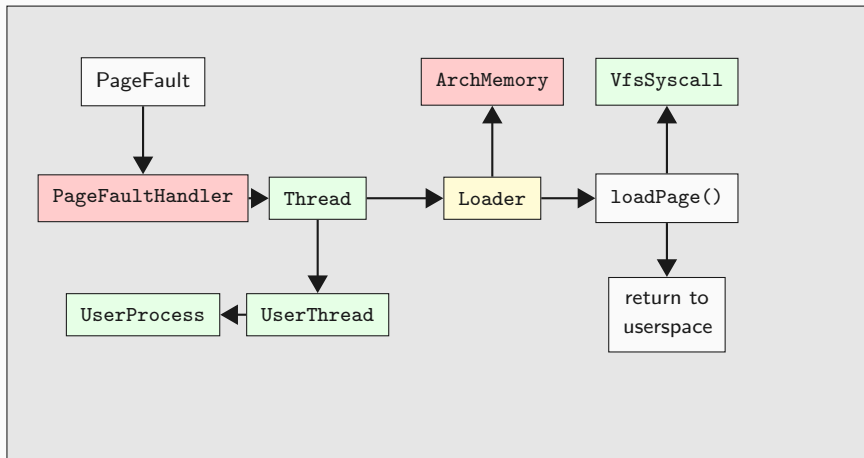


```
void ProcessRegistry::Run()
{
    debug(PROCESS_REG, "mounting userprog-partition \n");
    // ...
    vfs_syscall.mount("idea1", "/usr", "minixfs", 0);
    debug(PROCESS_REG, "mount idea1\n");

    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, ...

- 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 ;)

Status quo

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

- Every $54.925439\mu s$
- Implemented in InterruptUtils.cpp

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

- 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

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

```
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 Kernighan, 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, 0, 0);  
}
```

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

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();  
    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.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++

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

```
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!

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


```
#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());
```

```
}
```

AANNND

IT'S GONE.

```
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_;  
}
```

Get in touch with the source code!