TEAM
RUBUS

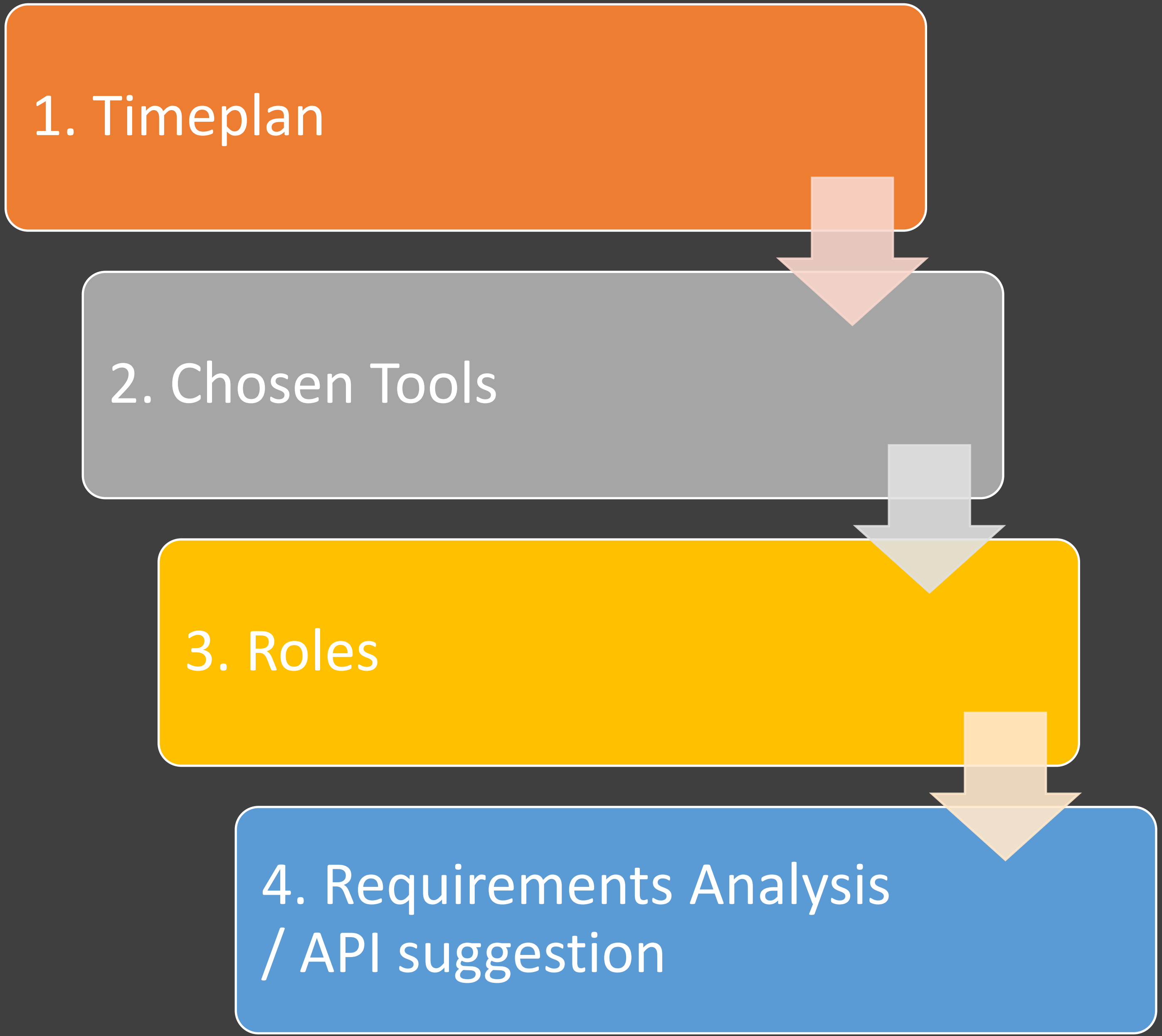
2nd Supervision Meeting

13.November.2020

Afram Afrem, Andreas Mäkilä, Victor Nicholas Ebirim, Lukas Johannes Dust, Aymen Nouidha

MÄLARDALEN UNIVERSITY
SWEDEN

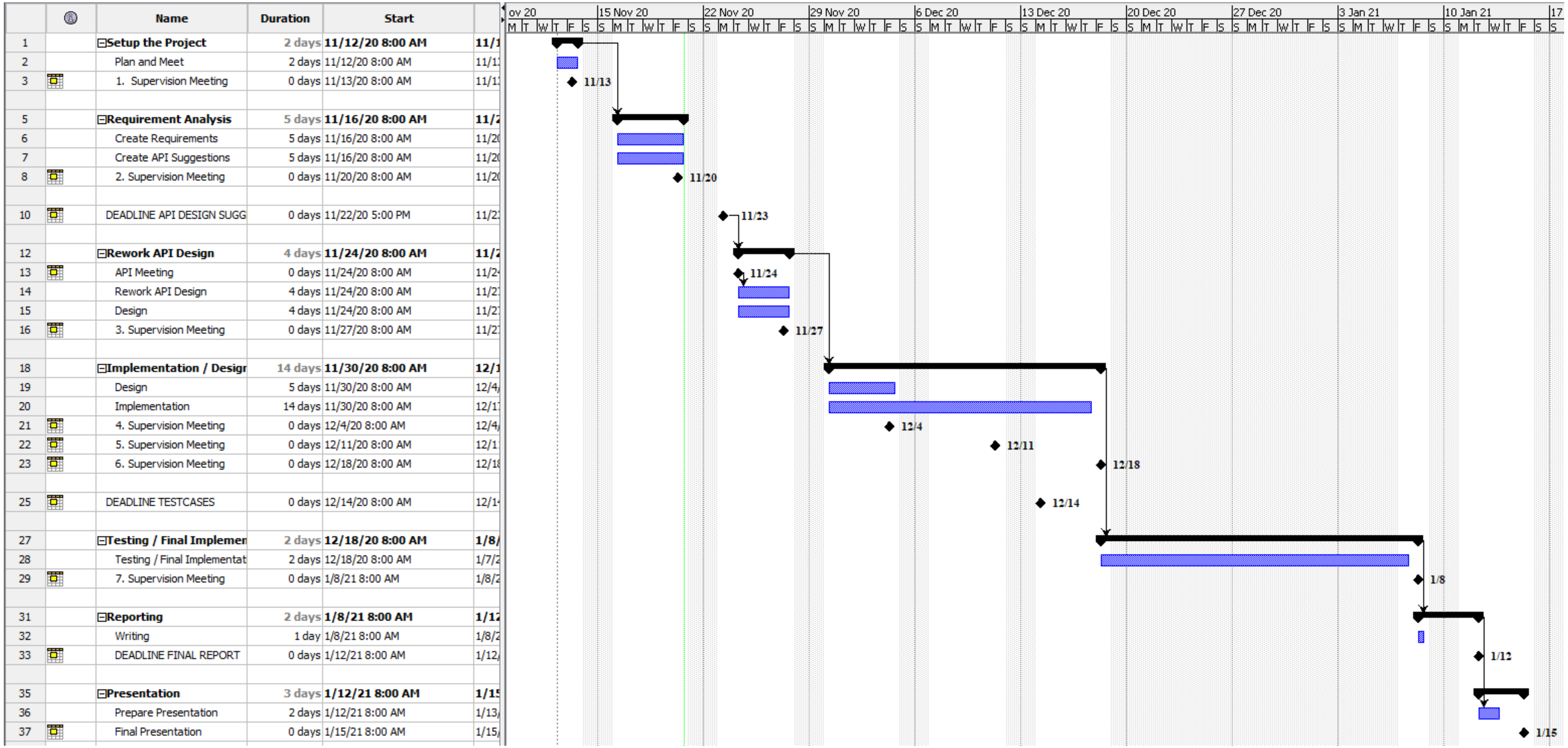
Agenda



1. Timeplan
2. Chosen Tools
3. Roles
4. Requirements Analysis / API suggestion

TEAM RUBUS

1. Timeplan



	ⓘ	Name	Duration	Start	
1		**⊟Setup the Project**	2 days	**11/12/20 8:00 AM**	**11/1**
2		Plan and Meet	2 days	11/12/20 8:00 AM	11/1
3	📅	1. Supervision Meeting	0 days	11/13/20 8:00 AM	11/1
5		**⊟Requirement Analysis**	5 days	**11/16/20 8:00 AM**	**11/2**
6		Create Requirements	5 days	11/16/20 8:00 AM	11/2
7		Create API Suggestions	5 days	11/16/20 8:00 AM	11/2
8	📅	2. Supervision Meeting	0 days	11/20/20 8:00 AM	11/2
10	📅	DEADLINE API DESIGN SUGG	0 days	11/22/20 5:00 PM	11/2
12		**⊟Rework API Design**	4 days	**11/24/20 8:00 AM**	**11/2**
13	📅	API Meeting	0 days	11/24/20 8:00 AM	11/2
14		Rework API Design	4 days	11/24/20 8:00 AM	11/2
15		Design	4 days	11/24/20 8:00 AM	11/2
16	📅	3. Supervision Meeting	0 days	11/27/20 8:00 AM	11/2
18		**⊟Implementation / Desigr**	14 days	**11/30/20 8:00 AM**	**12/1**
19		Design	5 days	11/30/20 8:00 AM	12/4
20		Implementation	14 days	11/30/20 8:00 AM	12/1
21	📅	4. Supervision Meeting	0 days	12/4/20 8:00 AM	12/4
22	📅	5. Supervision Meeting	0 days	12/11/20 8:00 AM	12/1
23	📅	6. Supervision Meeting	0 days	12/18/20 8:00 AM	12/1
25	📅	DEADLINE TESTCASES	0 days	12/14/20 8:00 AM	12/1
27		**⊟Testing / Final Implemen**	2 days	**12/18/20 8:00 AM**	**1/8/**
28		Testing / Final Implementat	2 days	12/18/20 8:00 AM	1/7/
29	📅	7. Supervision Meeting	0 days	1/8/21 8:00 AM	1/8/
31		**⊟Reporting**	2 days	**1/8/21 8:00 AM**	**1/1**
32		Writing	1 day	1/8/21 8:00 AM	1/8/
33	📅	DEADLINE FINAL REPORT	0 days	1/12/21 8:00 AM	1/12/
35		**⊟Presentation**	3 days	**1/12/21 8:00 AM**	**1/1**
36		Prepare Presentation	2 days	1/12/21 8:00 AM	1/13/
37	📅	Final Presentation	0 days	1/15/21 8:00 AM	1/15/

2. Selected Tools

- Project Management / Timeplan / Todos
  - Leantime
  - Github Project
- Requirments / Testcases / Design
  - OSRMT
- Versioning
  - Git
- Reporting / Writing Reports
  - Overleaf -> Andreas is the host
- Programming
  - Tera Term (Terminal)
  - Atmel Studio
  - Local Computer of Andreas -> setup with RDP
- Communication
  - Virtual Whiteboard: Mural
  - Troubleshooting: Microsoft Teams
  - Talking/Meeting: Discord

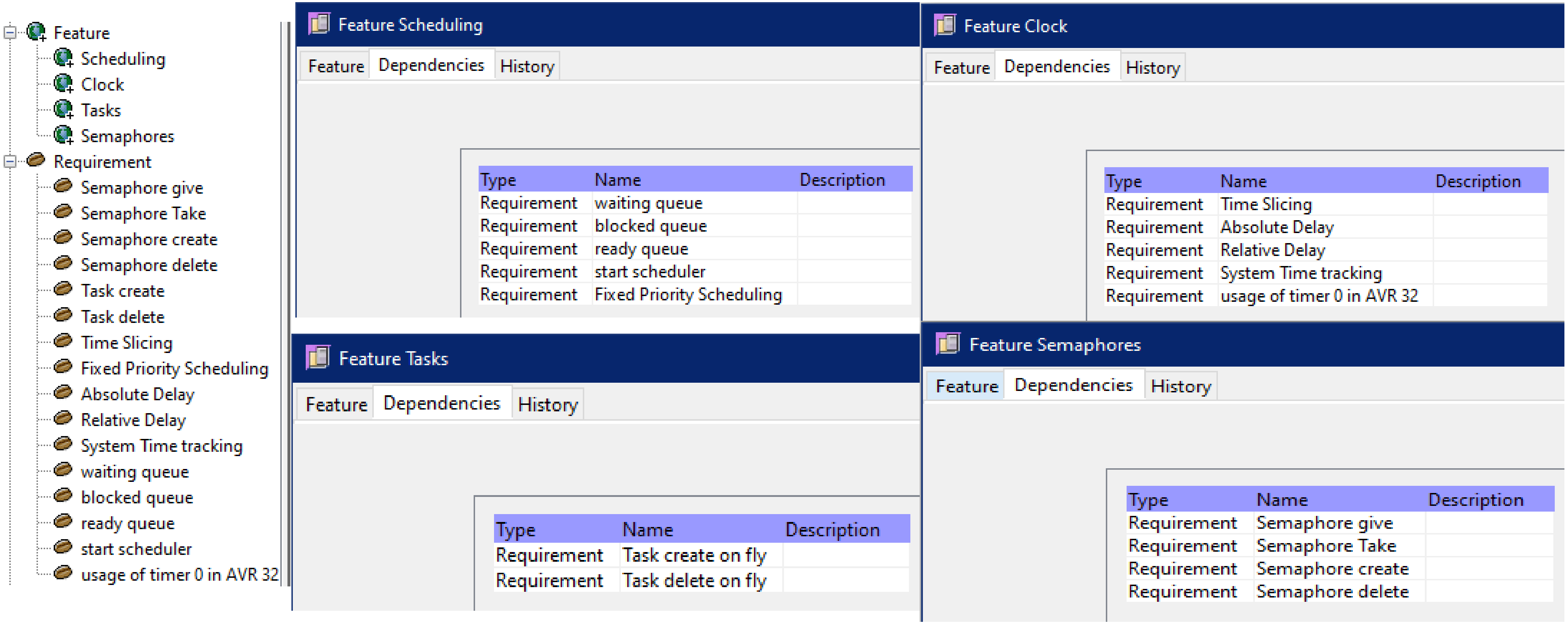
3. Roles and Rotation

Role	Responsibilities	Startperson	Rotation
Project Leader	Kanban Master, Schedule, keep track over the tools	Lukas Dust	None
Customer Contact	Contact to the Customer regarding requirements and questions Mailperson	Andreas Mäkilä	None
Developer		Afram Afrem	None
Developer		Victor Nicholas Ebirim	None
Developer		Aymen Nouidha	None

4. Requirements Analysis / API Design

Person	Assigned Task / Responsibility
Andreas Mäkilä	Semaphores
Aymen Nouidha	Scheduling
Victor Ebirim	Tasks
Lukas Dust and Afram Afrem	Clock

4. Requirements Analysis / API Design



- Feature
  - Scheduling
  - Clock
  - Tasks
  - Semaphores
- Requirement
  - Semaphore give
  - Semaphore Take
  - Semaphore create
  - Semaphore delete
  - Task create
  - Task delete
  - Time Slicing
  - Fixed Priority Scheduling
  - Absolute Delay
  - Relative Delay
  - System Time tracking
  - waiting queue
  - blocked queue
  - ready queue
  - start scheduler
  - usage of timer 0 in AVR 32

Feature Scheduling

Type	Name	Description
Requirement	waiting queue	
Requirement	blocked queue	
Requirement	ready queue	
Requirement	start scheduler	
Requirement	Fixed Priority Scheduling	

Feature Clock

Type	Name	Description
Requirement	Time Slicing	
Requirement	Absolute Delay	
Requirement	Relative Delay	
Requirement	System Time tracking	
Requirement	usage of timer 0 in AVR 32	

Feature Tasks

Type	Name	Description
Requirement	Task create on fly	
Requirement	Task delete on fly	

Feature Semaphores

Type	Name	Description
Requirement	Semaphore give	
Requirement	Semaphore Take	
Requirement	Semaphore create	
Requirement	Semaphore delete	

4.1. Aymen Nouidha / Scheduling

- Prerequisites for scheduling:
- Ready_queue
- Blocked_queue
  - When a task finishes executing it should be put back into the ready or blocked queue in the proper order depending on its order
- Task_PriorityAssignement(TCB* task, queue)
- TCB struct needs to be modified to include Period and Deadline??
  - →API Seminar
- The scheduling of tasks happens in the time interrupt, so we should modify it
  - Timer_ISR()
    - In this function we initialize and start scheduling
- start_Schedular()

4.2. Lukas Dust and Afram Afrem / Clock

- Prerequisites:
  - Timer initialization and creation is already implemented (AVR Timer 0)
  - Timer values (and Interrupt) needs to be initialized and can be used then
- Requirements:
  - Track the System time
  - Relative Delay Function
  - Absolute Delay Function
- API suggestion:
  - Int ROSA_getSystemTickCount(); → returns the system tick time –> API Seminar?
  - Void ROSA_sysTickWait(int ticks); → Relative Delay function
  - Void ROSA_sysTickWaitUntil(int Ticktime); → Absolute delay Function

4.3. Victor Ebirim / Tasks

- Requirements:
  - Dynamic creation and termination of tasks
- Prerequisites:
  - Static creation of tasks
- What do we need to implement?
  - We need to re-implement the task creation sub-routine to support dynamic creation
  - We need to implement a delete task function
  - We should include a task priority property on the tcb structure and also include as a parameter on the taskcreate function
  - We can implement the task suspension and activation

4.3. Victor Ebirim / Tasks

Suggested API

ROSA_tcbKill

Prototype:

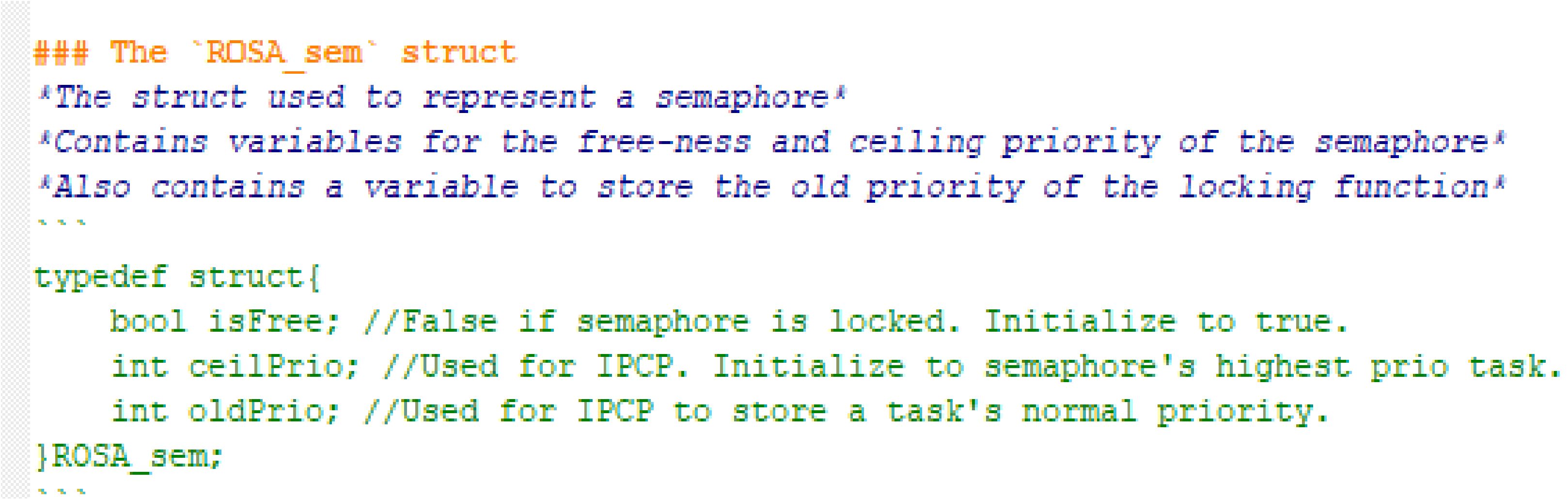
```
        unsigned char ROSA_tcbKill(tcb *TCB);
```

ROSA_start (re-implementation)

Prototype:

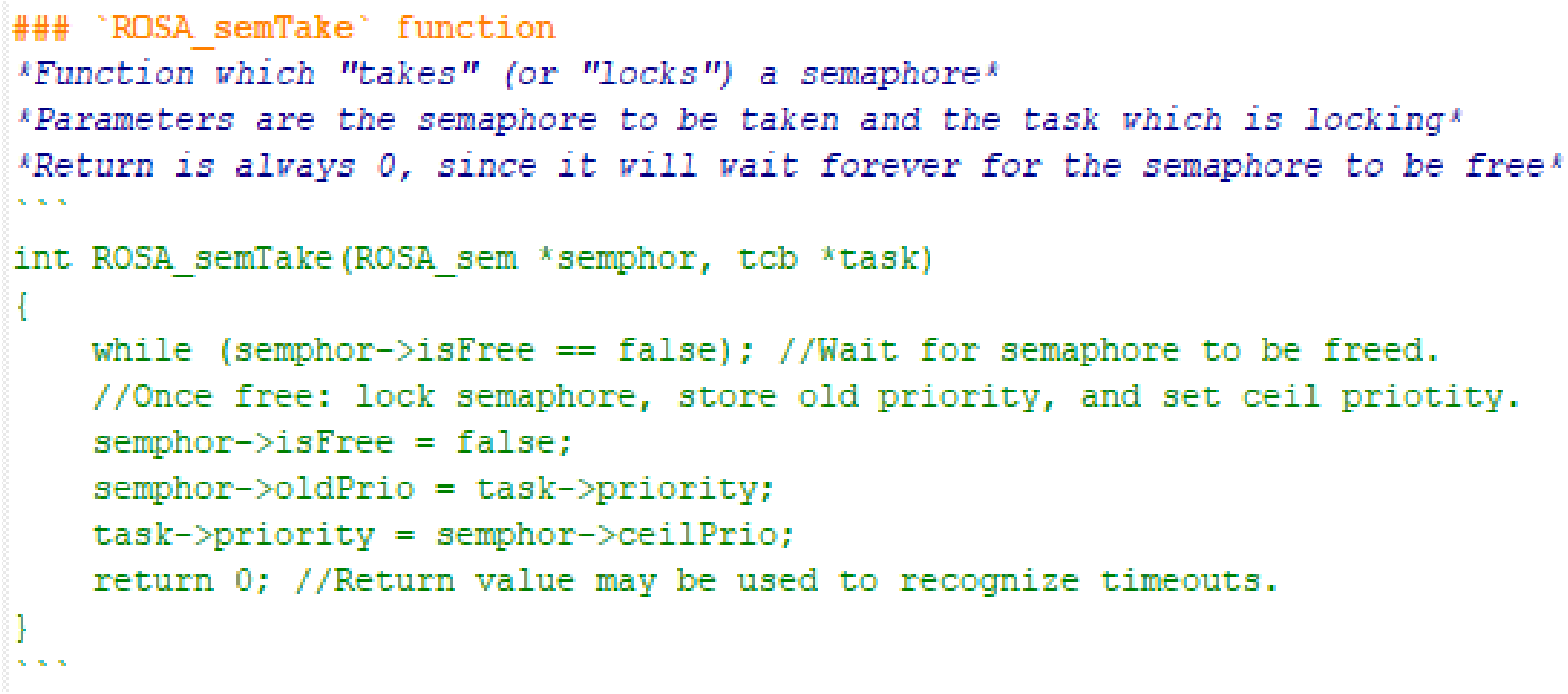
```
        void ROSA_start(tcb *TCB, char *id, void *taskFunc,
                unsigned char taskPrio, int *stack, int stackSize)
```

4.4. Andreas Mäkilä / Semaphores



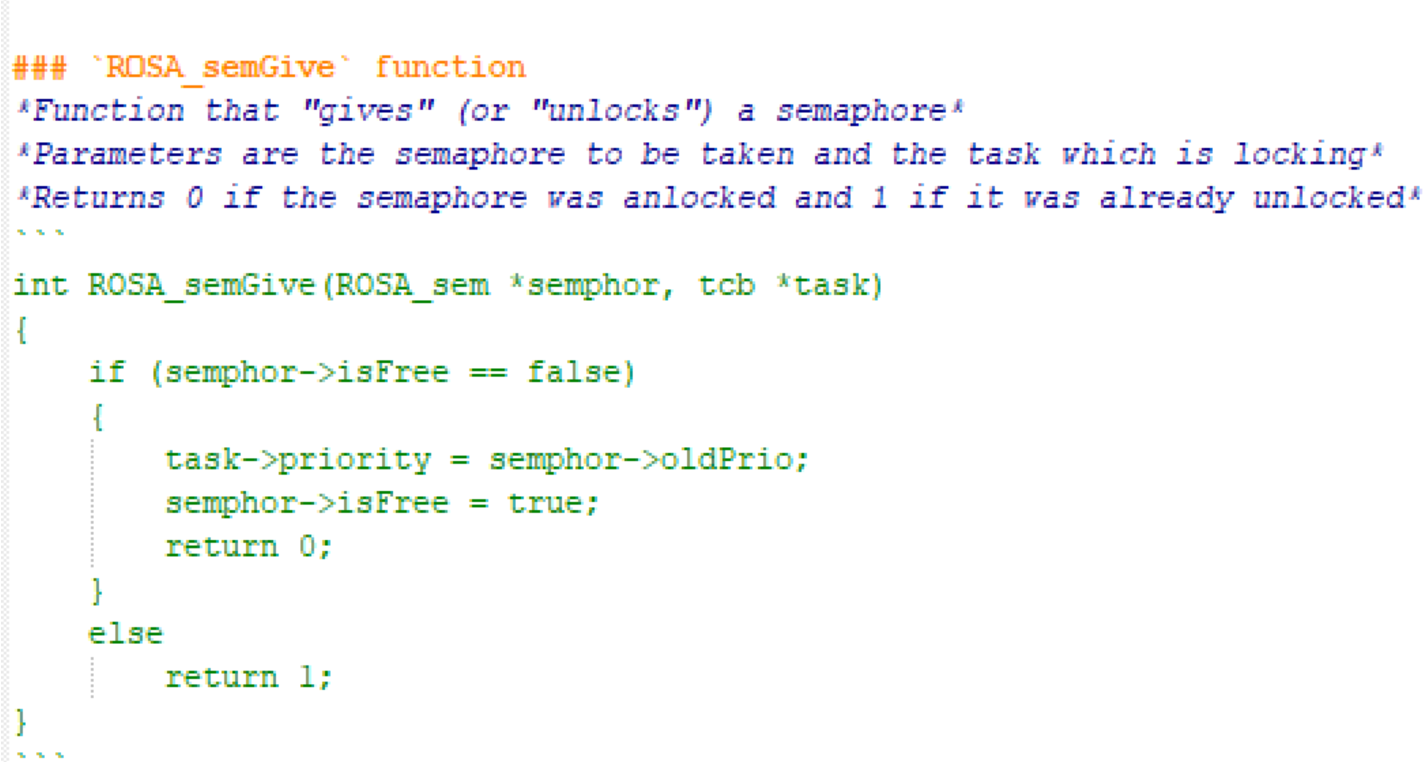
~~~~
### The `ROSA_sem` struct
*The struct used to represent a semaphore*
*Contains variables for the free-ness and ceiling priority of the semaphore*
*Also contains a variable to store the old priority of the locking function*
```
typedef struct{
    bool isFree; //False if semaphore is locked. Initialize to true.
    int ceilPrio; //Used for IPCP. Initialize to semaphore's highest prio task.
    int oldPrio; //Used for IPCP to store a task's normal priority.
}ROSA_sem;
```
~~~~

4.4. Andreas Mäkilä / Semaphores



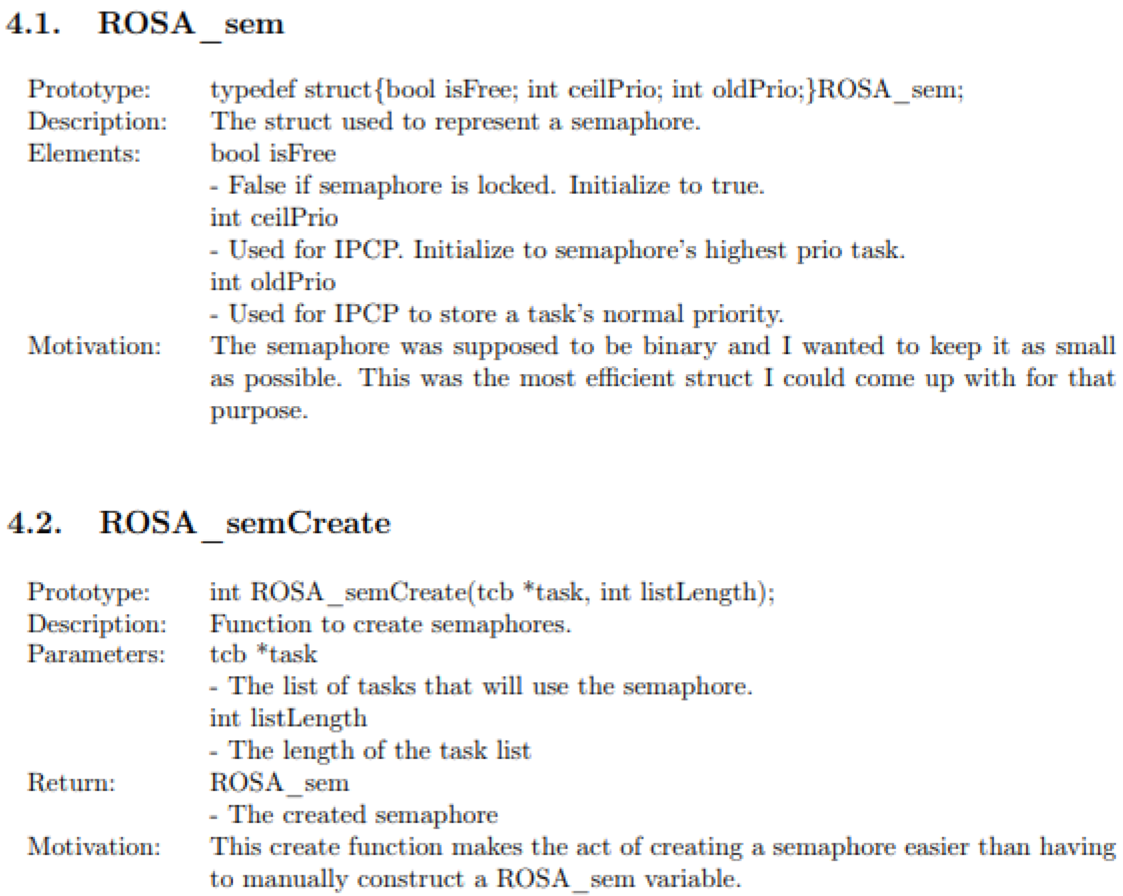
~~~~
### `ROSA_semTake` function
*Function which "takes" (or "locks") a semaphore*
*Parameters are the semaphore to be taken and the task which is locking*
*Return is always 0, since it will wait forever for the semaphore to be free*
```
int ROSA_semTake(ROSA_sem *semphor, tcb *task)
{
    while (semphor->isFree == false); //Wait for semaphore to be freed.
    //Once free: lock semaphore, store old priority, and set ceil priotity.
    semphor->isFree = false;
    semphor->oldPrio = task->priority;
    task->priority = semphor->ceilPrio;
    return 0; //Return value may be used to recognize timeouts.
}
```
~~~~

4.4. Andreas Mäkilä / Semaphores



~~~~
### `ROSA_semGive` function
*Function that "gives" (or "unlocks") a semaphore*
*Parameters are the semaphore to be taken and the task which is locking*
*Returns 0 if the semaphore was anlocked and 1 if it was already unlocked*
```
int ROSA_semGive(ROSA_sem *semphor, tcb *task)
{
    if (semphor->isFree == false)
    {
        task->priority = semphor->oldPrio;
        semphor->isFree = true;
        return 0;
    }
    else
        return 1;
}
```
~~~~

4.4. Andreas Mäkilä / Semaphores



4.1. ROSA_sem

Prototype: typedef struct{bool isFree; int ceilPrio; int oldPrio;}ROSA_sem;
Description: The struct used to represent a semaphore.
Elements: bool isFree
- False if semaphore is locked. Initialize to true.
int ceilPrio
- Used for IPCP. Initialize to semaphore's highest prio task.
int oldPrio
- Used for IPCP to store a task's normal priority.
Motivation: The semaphore was supposed to be binary and I wanted to keep it as small as possible. This was the most efficient struct I could come up with for that purpose.

4.2. ROSA_semCreate

Prototype: int ROSA_semCreate(tcb *task, int listLength);
Description: Function to create semaphores.
Parameters: tcb *task
- The list of tasks that will use the semaphore.
int listLength
- The length of the task list
Return: ROSA_sem
- The created semaphore
Motivation: This create function makes the act of creating a semaphore easier than having to manually construct a ROSA_sem variable.