# Real Time Operating Systems

## Scripting and programming in the Linux system

## C language

1. Create a data structure that contains the information about a task
   1. Task name
   2. Task priority
   3. Total task time
   4. Remaining task execution time

If you do not know how to do it, start from getting familiar with structures and pointers in C. To compile a C program use:

gcc program.c

To run the program:

./a.out

* 1. Create a new instance of task and show its contents with the following format

[name] - [priority] priority [total - remaining] / [total] done

eg.

Learn C - High priority 1 / 10 done

* 1. **What is the difference between Task \* task and Task task?**

1. Create a function that will initialize and return the new task. The signature might be:

Task \* createTask(char \*name, char \* priority,

int time, int remaining)

so it could be used:

Task \* task = createTask("Learn C", "High", 10, 9);

1. Create a function that will show the task in the above format. The signature might be:

void show(Task \* task)

Create a few tasks and show them using the show function.

1. Create a function that will decrease and increase the task time by one
   1. **What is the difference between passing the Task \* task and Task task as an argument of the function eg.**

void increase(Task \* task)

{

task->taskTime++;

}

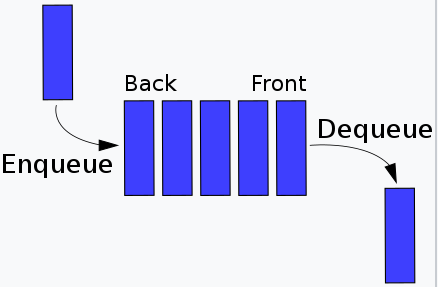
void decrease(Task task)

{

task.taskTime--;

}

1. Get familiar with the data structure called a queue. Then write a program in C/C++ that simulates the operation of a queue (without using the built in queue). The program should allow:
   1. Adding elements to the end of the queue
   2. Removing elements from the front of the queue
   3. Displaying the contents of the queue.



źródło: <https://en.wikipedia.org/wiki/Queue_(abstract_data_type)>

The queue should be composed of Nodes:

typedef struct Node {

int value;

struct Node\* next;

} Node;

The access to the front and the end of the queue should be made through the instance of the structure:

typedef struct Queue {

Node\* front;

Node\* back;

} Queue;

1. Extend the implementation of the queue to handle Tasks instead of integers
   1. correct the add, remove and display function
2. Create two queues:
   1. queue for important tasks
   2. queue for other tasks

Create 6 tasks with varying priorities and add them to the appropriate queues. Show the contents of the queues after the operations.