AGH	*Akademia Górniczo-Hutnicza w Krakowie*		
Faculty: ESA	Academic Year 2019/2020	Year of study	Fields of science:
Student name:	**Ioannis Manousaridis**		
Course:	**Real Time Operating Systems in Control Applications**		
Exercise nr:	**009P**		
Exercise subject:	**Exercise SOCR-09P. FreeRTOS – tasks priorities**		
Lecturer: dr inż. Grzegorz Wróbel			
Date: 00.00.2020			Grade:

1 Exercise purpose

The purpose of the exercise is to become familiar with the scheduler mechanisms and task prioritization.

2 Assumptions / Theory

In FreeRTOS each task is assigned a priority from 0 to (configMAX_PRIORITIES – 1), where configMAX_PRIORITIES is defined within FreeRTOSConfig.h. The lowest priority is the tskIDLE_PRIORITY or 0. The higher the number of priority, the more important is the task. The FreeRTOS scheduler ensures that tasks in the Ready or Running state will always be given processor (CPU) time in preference to tasks of a lower priority that are also in the ready state. In other words, the task placed into the Running state is always the highest priority task that is able to run.

3 Description of implementation

Two files were created in the scope of this exercise. The lab2_008.c is about the project 008-Priorities_1 and the lab2_009.c is about project 009-Priorities_2.

In the first file, lab2_008.c, the user has three options about creating two tasks. The first option will create two continuous tasks with the same priorities. The second will create two continuous tasks with the same priorities, and then it will change the priority of one task after some time. The last one will create two continuous tasks with different priorities. Depending of the input of the user an if case will select the appropiate commands to execute and create the tasks with the right prorities.

In the second file, lab2_009.c, the user has two options. The first is to run a continuous task and then run the next higher priority task created by the first task. The second is to

run a periodical task with a delay in the loop, and then from this task to run another periodical task with a higher priority with a delay in the loop.

In the first case, a continuous task is created (there is no delay in the loop). This task will create the second task, which is also continuous and when it is finished it will call the vTaskEndScheduler() and will terminate the program.

In the second case, the task procedure is the same, though this time the tasks are periodically (they have delays inside the loops) and they continue to execute for ever instead of calling the vTaskEndScheduler() function.

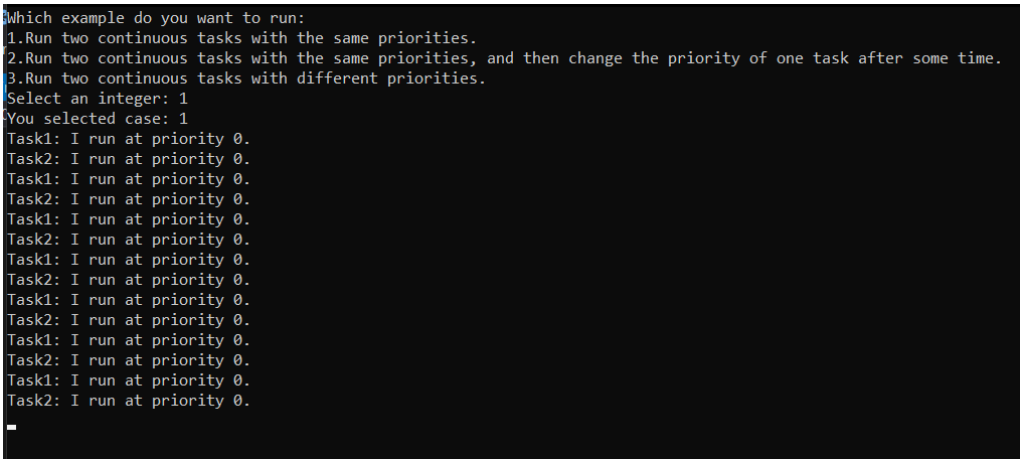

Figure 1: Two tasks running with the same priorities.

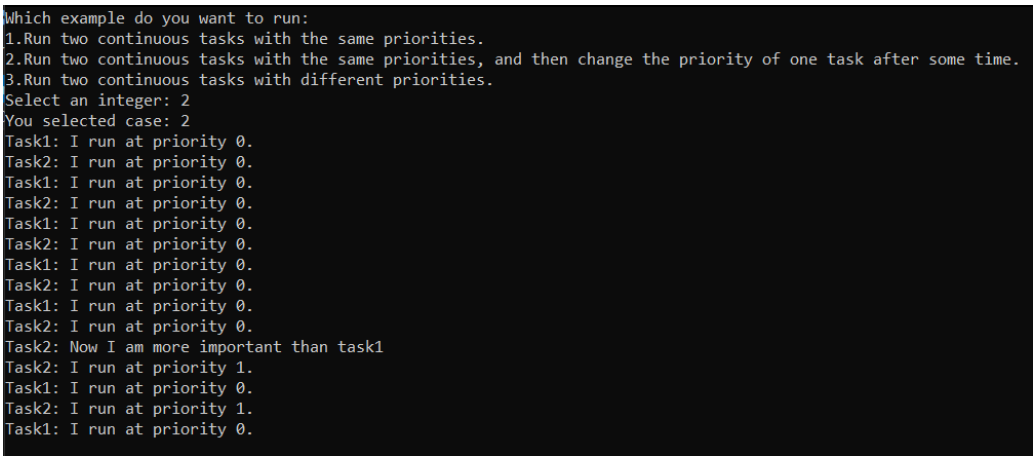

Figure 2: Two tasks running with the same priorities and task 2 changing its priority after 5 repeatitions.

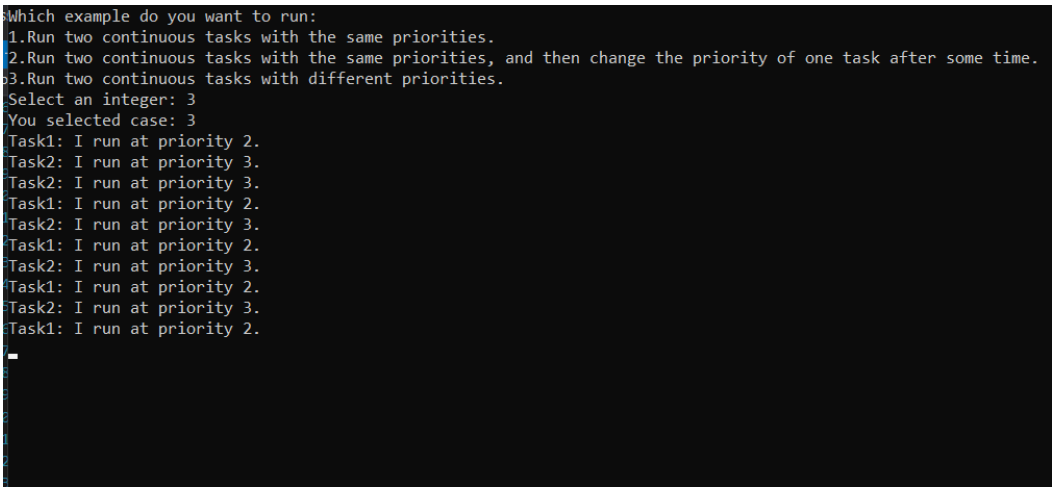

Figure 3: Two tasks running with the different priorities.

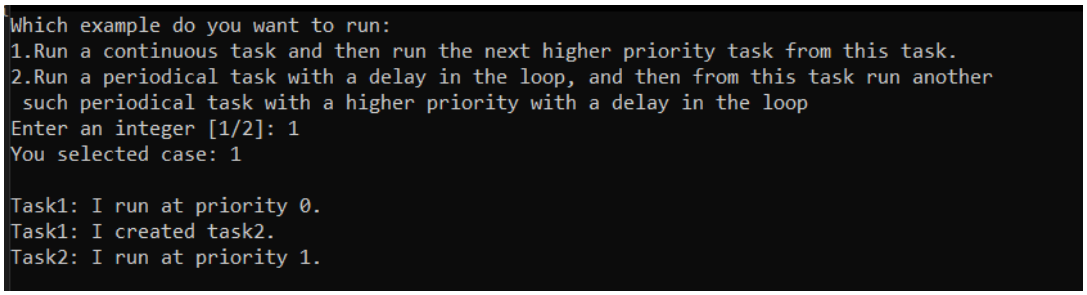

Figure 4: A continuous task creating another continuous task. Then the sceduler stops executing.

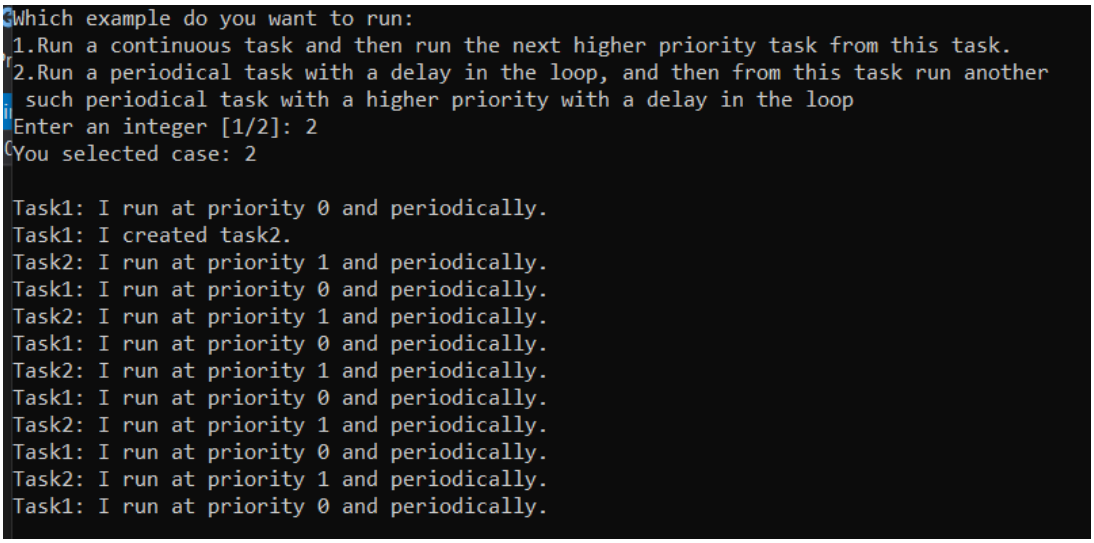

Figure 5: Two tasks running periodically with different priorities. Task 1 is responsible for the creation of Task 2.

4 Conclusions

In general, the FreeRTOS is a pretty clear OS depending of how it treats the tasks. Each task a priority which range from 0 to 4. The task with the highest priority is the which will execute first if there are more than one tasks executing at the same time. Also the FreeRTOS provides the possibility of changing the tasks' priorities.