From theory to practice: exploring FreeRTOS with QEMU

Computer Architectures and Operating Systems course

Alessandro Genova, Alessandro Torrisi, Giorgia Moscato, Simone Sambataro, Emanuele Cornaggia

Academic year: 2023-2024

4

3

2

1

OVERVIEW

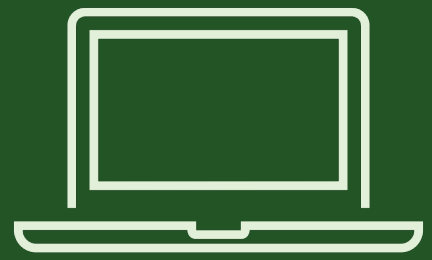
4

3

2

1

Creation of a step-by step guide that explains how to start using FreeRTOS on QEMU and to enable everyone to replicate the work we have done.



Implemention of pratical examples:

4

3

2

1

UART command line

LEDs Animation

Memory WatchDog

Communication between tasks

SEMAPHORE

NOTIFICATION

Implemention of new features:

4

Scheduling algorithm:

Non-preemptive: **S**hortest **J**ob **F**irst and **L**ongest **J**ob **F**irst
Preemptive: **E**arly **D**eadline **F**irst

3

2

1

Memory management algorithm:

Worst-fit
Best-fit

freeRTOS

Evaluation of performance for the implemented features

Worst Fit and Best Fit

EDF and Round Robin

UART command line

```
Received string: hello world
Received string: i am writing
Received string: into
Received string: uart
Received string: this is
Received string: a commandline
Received string: :)
```

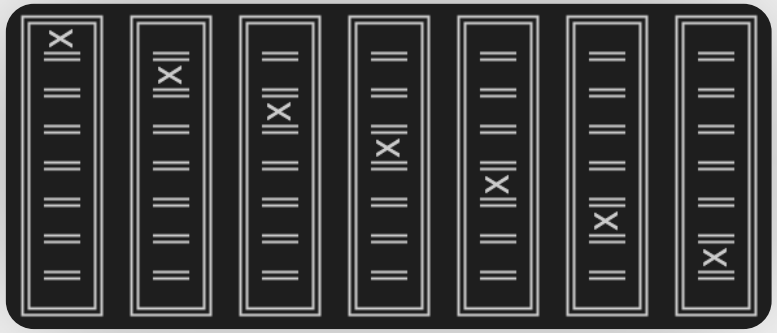
- Polling method
- Interrupt method
- Tickless IDLE
- Less power consumption

CPU Usage Statistics

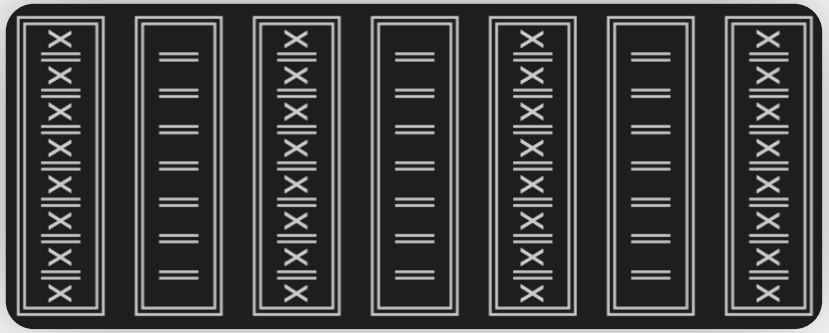
```
Received string: stats

Task name       Run time        Percentage      Task state
Commandline     97              <1%             Running
IDLE            379426          75%             Ready
Led Task        126435          24%             Suspended
Tmr Svc         7               <1%             Blocked
```

- Percentage of usage
- Task State
- Debug with stats flag
- Hardware timer

Led Animation



Knight Rider Effect



Constant Blink Effect

Notifications in FreeRTOS

```
The value of temperature is 25
The value of humidity is 85
The humidity has exceeded the threshold
 | | | |*|*|*|*
```

```
The value of temperature is 48
The value of humidity is 65
The temperature has exceeded the threshold
*|*|*|*| | | |
```

- Lightweight inter-task communication
- Simulation of two sensors: temperature and humidity
- Two threshold values
- If respective value is exceeded, half of leds light on

Memory WatchDog

```
Task 1 is running
Free heap statistics for task 'Task1':
 UsedHeapSpaceInBytes: 1888
 AvailableHeapSpaceInBytes: 59552
 SizeOfLargestFreeBlockInBytes: 59488
 SizeOfSmallestFreeBlockInBytes: 64
 NumberOfFreeBlocks: 2
 MinimumEverFreeBytesRemaining: 59488
 NumberOfSuccessfulAllocations: 48
 NumberOfSuccessfulFrees: 2
```

- Memory event-handler
- Check if thresholds exceeds
- If memory allocated by the task is too high -> terminates

Semaphore Example

```
im the producer, writing 1 on the buffer
im the consumer, printing the buffer value: 1
im the producer, writing 2 on the buffer
im the consumer, printing the buffer value: 2
im the producer, writing 3 on the buffer
im the consumer, printing the buffer value: 3
im the producer, writing 4 on the buffer
im the consumer, printing the buffer value: 4
im the producer, writing 5 on the buffer
im the consumer, printing the buffer value: 5
qemu-system-arm: terminating on signal 2
```

- Producer and Consumer Task
- Synchronization between two tasks

Non preemptive Scheduling Algorithms

- Shortest Job First
- Longest Job First

While FreeRTOS uses as base non preemptive algorithm First Come First Served.

	FIRST COME FIRST SERVED (FCFS)	SHORTEST JOB FIRST (SJF)	LONGEST JOB FIRST (LJF)
Average Waiting Time(unit)	324	173	389
Average Turnaround Time(unit)	382	218	468

Preemptive Scheduling Algorithm - EDF

```
Task3 start time: 8651
Task3 end time: 8662
Task3 elapsed time: 11
Task8 start time: 8663
Task8 end time: 8697
Task8 elapsed time: 34
Task5 start time: 8697
Task5 end time: 8759
Task5 elapsed time: 62
```

```
Task3 has respected the deadline
Task4 has respected the deadline
Task5 has respected the deadline
Task6 has respected the deadline
Task7 has respected the deadline
Task8 has respected the deadline
```

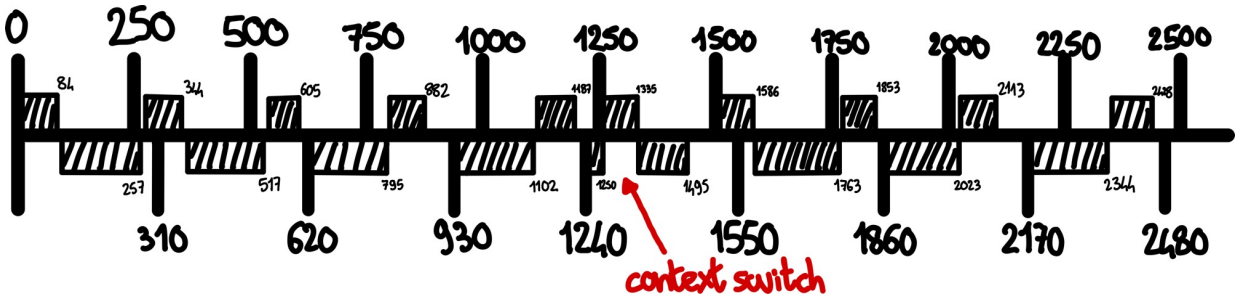
- Dynamic priority based on deadlines
- First execute task with the earliest deadline
- Check of missed deadlines
- Context switching based on deadline

Evaluation of EDF

	Round Robin	EDF
Percentage of respected deadlines	30%	88%

Comparison with Round Robin

	C	T
Task 1	95	250
Task 2	180	310



Gantt Diagram for two periodic tasks

Memory Management Algorithm

- Best-Fit
- Worst-Fit

```
Allocated block 10 of size 1378
Deallocated block 10
Allocated block 13 of size 2209
Allocated block 14 of size 4724
Allocated block 16 of size 3396
Allocated block 17 of size 1423

Can't allocate, block of size 1744 will overflow watchdog threshold.
Allocated block 19 of size 1732
Free block list:
Block 0: Address=536872216, Size=2920
Block 1: Address=536895312, Size=37568

Fragmentation is: (40488-37568)/(40488)
```

Blocks with random size

```
Fragmentation is: (60392-58824)/(60392)
Malloc of 128
Free block list:
Block 0: Address=536872216, Size=1568
Block 1: Address=536874200, Size=58680

Fragmentation is: (60248-58680)/(60248)
Freeing the 256 block
Free block list:
Block 0: Address=536872216, Size=1840
Block 1: Address=536874200, Size=58680

Fragmentation is: (60520-58680)/(60520)
Malloc of 64
Free block list:
Block 0: Address=536872216, Size=1840
Block 1: Address=536874280, Size=58600
```

Blocks with fixed size

Memory Management Algorithm

Evaluation of performance in two different cases

Table 1: Feasibility Percentage for Best Fit and Worst Fit Algorithms: random size test

Iteration	Worst Fit	Best Fit
1	7.0%	1.7%
2	8.0%	2.7%
3	20.0%	13.3%
4	0.4%	0.4%
5	10.0%	7.9%
6	4.0%	6.1%
7	15.0%	0.7%
8	12.8%	1.2%
9	0%	0%
10	8.0%	3.1%

Table 2: Fragmentation Percentage for Best Fit and Worst Fit: fixed size test

Iteration	Worst Fit	Best Fit
1	0%	0%
2	0%	0%
3	0%	0%
4	17.0%	17.0%
5	17.0%	1.2%
6	2.6%	0%
7	2.6%	0%
8	3.0%	0.4%
9	3.0%	0.3%
10	3.3%	0%

- For the Best Fit we have an average fragmentation of **3%** while for the Worst Fit we have a average fragmentation of **9%** (table 1)
- For the Best Fit we have an average fragmentation of **2%** while for the Worst Fit we have a average fragmentation of **5%** (table 2)

THANK YOU FOR YOUR ATTENTION!