Escuela de Informática, UNA.
EIF400 Paradigmas de Programación II-2017.
Carlos Loría-Sáenz.

Ejercicio Grupal de JS: Asincronía+Loop de Eventos.

PREGUNTAS

Este es un ejercicio de análisis teórico. Usamos `ES6/Node`. Considere y estudie el código adjunto (directorio `work`). Explique en cada caso las salidas mostradas de manera fundamentada. Requiere conocer la mecánica del loop de eventos. Se debe explicar el orden de los eventos. El caso 2 requiere un archivo en `data` de al menos 361MB llamado `huge`. Se puede crear con `medium`.

1. Caso `test_wait "small"`

```
PP:node test_wait.js small
>>> readFile scheduled ../data/small  at Wed Aug 30 2017 18:25:45 GMT-0600 (Hora estándar, América Central)
../data/small was read at Wed Aug 30 2017 18:25:45 GMT-0600 (Hora estándar, América Central)
>>> File: ../data/small  -> 355 bytes were read Wed Aug 30 2017 18:25:45 GMT-0600 (Hora estándar, América Central) <<<
>>> Starts: Async-1 at Wed Aug 30 2017 18:25:46 GMT-0600 (Hora estándar, América Central)
>>> Ends  : Async-1 at Wed Aug 30 2017 18:25:51 GMT-0600 (Hora estándar, América Central)
```

2. Caso `test_wait "huge"`

```
PP:node test_wait.js huge
>>> readFile scheduled ../data/huge  at Wed Aug 30 2017 18:27:46 GMT-0600 (Hora estándar, América Central)
>>> Starts: Async-1 at Wed Aug 30 2017 18:27:46 GMT-0600 (Hora estándar, América Central)
>>> Ends  : Async-1 at Wed Aug 30 2017 18:27:51 GMT-0600 (Hora estándar, América Central)
../data/huge was read at Wed Aug 30 2017 18:27:51 GMT-0600 (Hora estándar, América Central)
>>> File: ../data/huge  -> 369067631 bytes were read Wed Aug 30 2017 18:27:51 GMT-0600 (Hora estándar, América Central) <<<
```

3. Caso `timerProm 1 2`

```
PP:node timerProm.js 1 2
{ p1: 1, p2: 2, p3: undefined, p4: undefined }
then 1 666
sto 2 999

PP:node timerProm.js 1 2
{ p1: 1, p2: 2, p3: undefined, p4: undefined }
sto 2 999
then 1 666
```

4. Caso `timerProm 1 2 3 4`

```
PP:node timerProm.js 1 2 3 4
{ p1: 1, p2: 2, p3: 3, p4: 4 }
nt 4 777
sto 2 999
then 1 666
simm 3 888
```

5. Caso nextTick

```
PP:node nextTick.js

*** 1) MAIN 1: On top of the main script ***
1) start
1) end
1) nextTick
1) timeout
1) immediate

PP:node nextTick.js  2

*** 2) MAIN 2: Inside I/O callback ***
2) start
2) end
2) nextTick
2) immediate
2) timeout
```