Carrera de Especialización en Sistemas Embebidos
Sistemas Operativos en Tiempo Real 2.

Autor: Alejandro Celery
Fecha de creación: 14/09/2018

Práctica de drivers asincrónicos:

El objetivo de esta práctica es aprender a manejar operaciones de I/O usando drivers asincrónicos aptos para su uso en sistemas de tiempo real.

Este tipo de drivers conlleva una complejidad superior a los drivers sincrónicos pero son deseables dado que facilitan el cumplimiento de compromisos de tiempo real.

Descripción conceptual de la práctica:

Se propone caracterizar la performance y uso de recursos del servidor planteado en la práctica (1). Para eso se implementará el patrón *asynchronous completion token*.
Esto implica asociar un token a los paquete recibidos que indiquen la operación “medir performance” y que lo acompañe durante todo su tráfico dentro del sistema, llevando cuenta de las distintas métricas.

Arquitectura de la aplicación:

Se mantendrá la arquitectura de la práctica (1) y se agregará lo necesario para implementar el patrón ACT.

- Se reemplazarán las colas “queMaysuculizados” y “queMinusculizados” por una única cola “queTransmisión”.
- Se agregará un completion handler de “se terminó de enviar un paquete a medir perfomance”.
- El token se ubicará en memoria allocada de un pool de tamaño acorde.
- Esta tarea convertirá SIEMPRE a mayúsculas y mientras lo hace irá tomando las métricas necesarias.
    - Esto lo hará con el mismo algoritmo que la tarea “queMaysculizar”.
- Al acabar pondrá un puntero al paquete en la cola de salida.
    - También registrará un “completionHandler” de la operación para que sea llamado al terminar la transmisión y así poder marcar el tiempo de fin.
- Luego generará un paquete informativo de la performance y lo pondrá en la cola de salida también.

Paquetes de datos: **(en negrita los campos diferentes de la práctica 1)**

1B	1B	1B	T BYTES	1B
STX	OP	T	DATOS	ETX

Delimitación de paquete:

STX: Carácter 0x55
ETX: Carácter 0xAA.

Campos:

OP (Operación):
0: Convertir los datos recibidos a mayúsculas.
1: Convertir los datos recibidos a minúsculas.
2: Reportar stack disponible.
3: Reportar heap disponible.
4: Mensajes de estado de la aplicación.
5: Medir performance del sistema.

T (Tamaño): El tamaño del texto recibido.
DATOS: Texto a procesar. Deben ser caracteres ASCII legibles.

Requerimientos de software: (adicionales a los de la práctica 1)

R1: Cada paquete indicado para medir performance llevará un token de acuerdo al patrón ACT.
R1.1: El token llevará un campo uint32_t id_de_paquete que identificará un paquete durante su tránsito por el sistema.
R1.2: El campo id_de_paquete será 0 para el primer token que procese el sistema
R1.3: El campo id_de_paquete se incrementará en uno por cada token que se procese.
R1.4: El campo uint8_t * payload apuntará al paquete de datos a procesar.
R1.5: Los tokens se allocarán de un pool de memoria.

R2: Los punteros a token asociados a mediciones de performance se recogerán de la cola “queMedirPerformance”.

R3: Los paquetes indicados para medir performance se transmitirán por la cola “queTransmision” igual que los paquetes normales.

R4: Se reemplazarán las múltiples colas de transmisión de paquetes por una única cola llamada “queTransmision”.

R5: La aplicación medirá el tiempo de transmisión de los paquetes de datos y los informará por el puerto serie.
R5.1: uint32_t tiempo_de_llegada: tiempo en que se recibió el primer byte del paquete (el carácter STX).
R5.2: uint32_t tiempo_de_recepcion: tiempo en que se recibió el último byte del paquete (el carácter ETX).

R5.3: uint32_t tiempo_de_inicio: tiempo en que se extrae el puntero al paquete en la tarea “mayusculizar” o “minusculizar”.
R5.4: uint32_t tiempo_de_fin: tiempo en que se pone el puntero al paquete en la tarea “queMayusculizados” o “queMinusculizados”.
R5.5: uint32_t tiempo_de_salida: tiempo en que se transmitió el primer byte del paquete (el carácter STX).
R5.6: uint32_t tiempo_de_transmision: tiempo en que se transmitió el último byte del paquete (el carácter ETX).
R5.7: Todas los tiempos se medirán con el timer del RTOS usado.

R6: La aplicación medirá el consumo de memoria asignado a cada paquete junto con su tamaño recibido.
R6.1: uin16_t largo_del_paquete: largo total del paquete recibido (largo de datos MÁS largo del header).
R6.2: uin16_t memoria_alojada: tamaño del bloque extraído del pool de memoria del sistema para el bloque.

Requerimientos opcionales:

R7: Los tiempos se medirán con un timer de hardware dedicado con resolución 1us.

Sugerencias

- Usar una máquina de estados para tomar las mediciones que corresponda en el momento que corresponda.
- Se pueden agregar variables al token, los requerimientos no indican “agregue SOLO estas”.
- El mecanismo para medir performance no está explicitado en los requerimientos, se puede elegir cualquiera mientras que se tomen todas las mediciones correctamente.
- Una opción puede ser crear una nueva tarea “taskMedirPerformance” que haga el mismo proceso que “taskMayusculizar” pero que además intercale llamados a “fsmMedirPerformance (Token * t)” en los lugares necesarios. Si se agrega una variable de estado al token se la puede usar para saber qué medición tomar en cada momento.

Historial de cambios

17/09/2018 - ACelery: Se agregan los requerimientos funcionales.