Demostración de Conocimientos

Introducción

Estimado postulante, recibe un cordial saludo.

Desde ELDAR queremos conocer un poco tus habilidades técnicas en el desarrollo de diversos escenarios.

En cada escenario el objetivo es que demuestres de qué manera podrías resolverlo.

Si algún tópico/escenario no se entiende, nos podés contactar mediante mail a las casillas: alejandra.estrella@eldars.com.ar y marcelo.lombardo@eldars.com.ar

Si algún tópico/escenario está fuera de tu conocimiento está bien.

Desde ELDAR creemos profundamente en el trabajo en equipo, en el acompañamiento, y el mentoring de las personas.

Es por eso que te estamos buscando.

Creemos en vos y sabemos que vas a dar lo mejor.

¡Te deseamos éxitos!

1. Cuéntanos acerca de tu experiencia

a. Has escuchado hablar o tienes experiencia en dispositivos con Software embebido?.
De ser así, cuéntanos que dispositivos has utilizado bajo esta modalidad o alguna experiencia personal.

b. Has escuchado hablar o tienes experiencia en puertos de comunicación?.
De ser así, cuéntanos que tipo de puertos conoces o alguna experiencia personal al respecto. Si conoces alguno de ellos, bríndanos una breve descripción de sus características y para qué los has utilizado.

c. Tuviste la oportunidad de analizar el contenido de un registro de datos a bajo nivel de información? (ASCII, HEXADECIMAL, BCD o Binario)?.
Si tuviste tal experiencia, describinos a que estaba orientado dicho registro de datos o alguna descripción que consideres importante.

2. Data Parsing – Analisis de registro de datos

Descripción del sistema:

Un dispositivo recibe una orden para moverse N pasos hacia adelante o N pasos hacia atrás.

Para lo cual, dicho dispositivo recibe desde una computadora un string de comando y luego de ejecutar el comando, el dispositivo responde a la computadora con un Respuesta.

Cada Comando enviado por la Computadora, tiene un Identificador de Comando, y la respuesta enviada por el dispositivo mantiene el mismo Identificador de Comando.

El dispositivo tiene la suficiente capacidad de: recibir el comando, ejecutar la orden, responder al comando y continuar así con el comando siguiente.

Los comandos enviados por la Computadora hacia el dispositivo y las respuestas enviadas por el dispositivo a la computadora, son almacenados de forma concatenada en un archivo LOG.

A continuación se detallan las especificaciones de Comandos y Respuestas:

La estructura de un comando es como se muestra:

RQ	ID	Dirección	N Pasos	FN

La estructura de una respuesta es como se muestra:

RT	ID	Resultado	FN

Tabla de Definiciones para el Comando:

Campo del Comando	Descripción	Formato	Scope de Datos	Longitud (Bytes)
RQ	Encabezado	ALFABETICO CONSTANTE	**SIEMPRE DEBE SER RQ**	2
ID	Identificador de Comando	NUMERICO VARIABLE	00 HASTA 99	2
Dirección	Sentido de Avance	ALFABETICO VARIABLE	AV = Avance RV= Retroceso	2
N Pasos	Cantidad de Pasos a ejecutar en la orden	NUMERICO VARIABLE	00-10	2
FN	Fin de Comando	ALFABETICO CONSTANTE	**SIEMPRE DEBE SER FN**	2

Tabla de Definiciones para la Respuesta:

Campo del Comando	Descripción	Formato	Scope de Datos	Longitud (Bytes)
RT	Encabezado	ALFABETICO CONSTANTE	**SIEMPRE DEBE SER RT**	2
ID	Identificador de Comando	NUMERICO VARIABLE	00 HASTA 99	2
Resultado	Código de Resultado	ALFABETICO VARIABLE	00 = Ejecutado 99= ERROR	2
FN	Fin de Comando	ALFABETICO CONSTANTE	**SIEMPRE DEBE SER FN**	2

A continuación se muestran dos lineas del Archivo LOG obtenido en la computadora:

RQ01AV10FNRT0100FNRQ02AV03FNRT0200FNRQ03RV03FNRT0300FNRQ04RV99FNRT0499FN

RQ05AV30FNRT0500FNRQ06AV30FNRT0699FN

Analizando las lineas de LOG, se pide:

a. Realizar el parsing (Separación) de Comandos y Respuestas. De la siguiente manera:

Comando01

Respuesta01

.

.

ComandoNN

RespuestaNN

b. Indicar el ID de la Respuesta que obtuvo Código de Error.

c. Indicar Por cada Comando leido en el LOG:

ID – Pasos Ejecutados – Dirección

d. Calcular la cantidad de Bytes en LOG.

e. De acuerdo a las Definiciones de Campos de Comando y Respuesta, qué tipo de control o validación de datos realizaría?. Ejemplifique solo 2 casos.

1. Cuéntanos acerca de tu experiencia

a) Tengo alguna experiencia con dispositivos de software embebido, aunque meramente académica en proyectos universitarios.
Por ejemplo, programamos un pequeño robot de movimiento en Arduino, o programamos la salida de una máquina expendedora en lenguaje assembler con un procesador MotorolaHc11, también programamos un sensor de iluminación en C, todas experiencias muy didácticas e interesantes que por desgracia no pude aplicar fuera del ámbito académico.

b) Aunque he escuchado, leído sobre ellos y conozco en su mayoría los principales puertos TCP/UDP para el movimiento de información en internet junto con su funcionalidad, lamentablemente nunca tuve la oportunidad de trabajar directamente controlando puertos de comunicación fuera de configuraciones estándar de aplicación web.

c) Solo trabajando en lenguaje assembler con unos pocos procesadores, ya que se manejan todos los registros de bit a bit a través de las funciones base del procesador, es lo normal en ese tipo de lenguajes trabajar en bajo nivel, lo mas destacable que uno aprende en ese tipo de lenguajes es a notar la importancia de la arquitectura de la cpu y como los registros deben variar en función de la misma, ya que una variación pequeña como el tamaño de un bus, o el cambio de lectura de Little endian a Big endian puede cambiar totalmente la performance de un programa.

2. Data Parsing – Analisis de registro de datos

Para esta prueba decidí usar JS, simplemente porque considero su biblioteca de string de las mas completas junto con la de PHP a diferencia de C o C++. Una biblioteca de string completa facilita el parseo de información en archivos de texto por lo que era una opción cómoda.

e) Viendo que es un archivo de registros de longitud fija, una validación rápida y sencilla pero efectiva seria ver que cada registro tenga su debido tamaño en cuanto falle ya sabrías que el registro esta corrupto o hubo fallos en el envió.
Otra verificación simple pero eficaz es aprovechar el formato de clave de inicio y clave de cierre para verificación, ya que si un registro no abre o cierra con su debida clave (RT FN) podrías detectar errores rápidamente.

<Codigo en Github click Aqui>

prueba.js | log.txt

prueba.js > fs.readFile("log.txt", 'utf8') callback

```
const fs = require('fs')
let log = ""
let logBytes=0;

fs.readFile("log.txt", 'utf8', (err, data) => {  //Leo el archivo Log.txt
    if (err) throw err;
    log += data;
    log = log.replace(("\n"),"") // Borro saltos de linea.
    log = log.replace(("\r"),"") // Borro saltos de linea.
    let arr = log.split("FN") //Uso el Marcador de fin para partir el string en cada comando y respuesta.
    arr = arr.filter(Boolean)
    console.log(arr)
    arr.map((e) => {
        if (e.slice(0, 2) == "RQ") {
            logBytes+=10;
            console.log("Comando" + e.slice(2, 4) + " " + e)
            console.log("ID: " + e.slice(2, 4))
            console.log("Pasos ejecutados: " + e.slice(6, 8))
            console.log("Direccion: " + e.slice(4, 6) + `(${e.slice(4, 6) == "AV" ? "Avance" : "Retroceso"})`)
        }
        if (e.slice(0, 2) == "RT") {
            logBytes+=8;
            console.log("Respuesta" + e.slice(2, 4) + " " + e)
            if (e.slice(4, 6) == "99") {
                console.log("Error en la respuesta con ID: " + e.slice(2, 4))
            }
            console.log("\n")
        }
    })
    console.log(` El numero de bytes del log es: ${logBytes}`)
    //Nota: no considero los saltos de linea como parte de los bytes del log, ya que no estaban en la tabla, pero bien deberian ser contados.
})
```

Salidas en consola

Nota: El enunciado no dejo claro si debía crear un nuevo archivo con los datos parseados, modificar el archivo existente o simplemente mostrar la nueva información por consola
Por lo que opte por solo mostrarlo por consola. En caso contrario no habría que añadir demasiado.

```
[
  'RQ01AV10', 'RT0100',
  'RQ02AV03', 'RT0200',
  'RQ03RV03', 'RT0300',
  'RQ04RV99', 'RT0499',
  'RQ05AV30', 'RT0500',
  'RQ06AV30', 'RT0699'
]
Comando01 RQ01AV10
ID: 01
Pasos ejecutados: 10
Direccion: AV(Avance)
Respuesta01 RT0100


Comando02 RQ02AV03
ID: 02
Pasos ejecutados: 03
Direccion: AV(Avance)
Respuesta02 RT0200


Comando03 RQ03RV03
ID: 03
Pasos ejecutados: 03
Direccion: RV(Retroceso)
Respuesta03 RT0300
```

```
Comando04 RQ04RV99
ID: 04
Pasos ejecutados: 99
Direccion: RV(Retroceso)
Respuesta04 RT0499
Error en la respuesta con ID: 04


Comando05 RQ05AV30
ID: 05
Pasos ejecutados: 30
Direccion: AV(Avance)
Respuesta05 RT0500


Comando06 RQ06AV30
ID: 06
Pasos ejecutados: 30
Direccion: AV(Avance)
Respuesta06 RT0699
Error en la respuesta con ID: 06


 El numero de bytes del log es: 108
```