PL4	01	Marques Ramos Francisco Mier Montoto	Marcel Juan
N.º PL	Equipo	Apellidos	Nombre

35625337-Q 71777658-V	UO289464@uniovi.es UO283319@uniovi.es
DNI	e-mail

1	Desarrollo de un inyector de carga	
N.º Práctica	Título	Calificación

Comentarios sobre la corrección

Asignatura de

CONFIGURACIÓN Y EVALUACIÓN DE SISTEMAS

Curso 2022-2023

Área de Arquitectura y Tecnología de Computadores
Departamento de Informática de la Universidad de Oviedo

Índice

Tabla de datos

Prueba	Concepto	Valor esperado	Valor obtenido
1	Media del tiempo de reflexión	~1 segundo (distr. exponencial 1seg)	1,07906s
	Número de peticiones de hilo	100 (100 pet/usr, 1usr/hilo)	100
2	Media de del tiempo de reflexión	~1 segundo (distr. exponencial 1seg)	1,07906s
	Número de peticiones por hilo	100 (100 pet/usr, 1usr/hilo)	100
3	Media del tiempo de reflexión	~1 segundo (distr. exponencial 1seg)	1,07906s
	Número de peticiones por hilo	100 (100 pet/usr, 1usr/hilo)	100
4	Media del tiempo de reflexión	~1 segundo (distr. exponencial 1seg)	1,07906s
	Número de peticiones por hilo	100 (100 pet/usr, 1usr/hilo)	100

Cuestionario del anexo a la práctica 1

1. **¿Todas las filas del archivo del registro contador de rendimiento son significativas? ¿Por qué?**

   Al principio y al final del archivo, se realizan mediciones durante los transitorios de entrada y de salida, por lo que las mediciones no son totalmente representativas del uso de recursos y del rendimiento real del sistema.

2. **¿Cuál es la utilización promedio del procesador? Explica cómo la has calculado.**

$$\%promedio\ de\ CPU = \frac{\sum \%tiempo\ de\ procesador\ en\ cada\ medición}{total\ de\ mediciones} \times 100$$

$$\%promedio\ de\ CPU = 43{,}071\%$$

3. **¿Cuál es la utilización promedio de la memoria? Explica cómo la has calculado.**

$$Bytes\ ocupados = Bytes\ instalados - Bytes\ disponibles - Bytes\ de\ caché$$

$$\%\ promedio\ de\ RAM = \frac{\sum Bytes\ ocupados\ en\ cada\ medición}{total\ de\ mediciones} \times 100$$

$$\%\ promedio\ de\ RAM = 22{,}703\%$$

4. **Como el experimento (inyector y servidor) se ejecuta en la misma máquina, no debe existir tráfico de red. ¿Cuál es el ancho de banda actual? ¿En qué unidades viene expresado? Suponiendo que el valor medio de contador Total de bytes / s fuera de 850000. ¿Cuál sería la utilización de la red? Explica cómo la has calculado.**

   Pese a que, durante la medición, el total de bytes por segundo no haya sido 0 constantemente, esto se puede atribuir a otros factores como peticiones del navegador en segundo plano. El ancho de banda está definido por la velocidad de la interfaz de red que se esté utilizando, en este caso 1Gbps.

   1Gbps = 1000000000 bits/s = 125.000.000 bytes/s $\rightarrow \frac{850.000}{125.000.000} \times 100 = 0{,}68\ \%$

Código fuente del inyector

```
// all lines of code have been manually coded.
// the structure resembles the original code structures provided, but nothing has been directly
copied.
// Juan Mier (UO283319) '22

#include <windows.h> // DWORD, HANDLE, WINAPI
#include <iostream> // cout, cerr, endl
#include <fstream> // save results to file
using namespace std; // this removes the need of writing "std::"" every time.

constexpr unsigned int MAXUSERS = 100;
constexpr unsigned int MAXPETITIONS = 100;
constexpr unsigned int PORT = 57000;
constexpr unsigned int PETITION_SIZE = 1250;
constexpr unsigned int RESPONSE_SIZE = 1250;
constexpr auto SERVERIP = "127.0.0.1"; // currently localhost

unsigned int totalUsers;
unsigned int petitionsPerUser;
float reflexTime;

typedef struct {
        int contPet;
        float reflex[MAXPETITIONS];
} threadParams;

threadParams threadInfo[MAXUSERS];

// ---

float GenerateRandomFloat(float lowerLimit, float upperLimit) {
        float num = (float)rand();
        num = num * (upperLimit - lowerLimit) / RAND_MAX;
        num += lowerLimit;
        return num;
}

float GenerateExponentialDistribution(float average) {
        float num = GenerateRandomFloat(0, 1);
        while (num == 0 || num == 1) {
                num = GenerateRandomFloat(0, 1);
        }
        return (-average) * logf(num);
}

void errorMessage(string message) {
        cerr << message << endl;
        exit(EXIT_FAILURE);
}

// ---

DWORD WINAPI Usuario(LPVOID parameter) {
        int threadNum = *((int*) parameter);
        char petition[PETITION_SIZE];
        char response[RESPONSE_SIZE];

        threadInfo[threadNum].contPet = 0;

        for (int i = 0; i < petitionsPerUser; i++) {
                //printf("[DEBUG] Peticion: %d, usuario: %d\n", i, numHilo);

                SOCKET s = socket(AF_INET, SOCK_STREAM, 0);

                if (s == INVALID_SOCKET) {
                        errorMessage("Ha ocurrido un error al inicializar el socket.");
                }

                sockaddr_in serv;
                serv.sin_family = AF_INET;
                serv.sin_addr.s_addr = inet_addr(SERVERIP);
                serv.sin_port = htons(PORT + threadNum);
```

```
		// connect
		if (connect(s, (struct sockaddr*) & serv, sizeof(serv)) == SOCKET_ERROR) {
			errorMessage("Error al conectar al servidor.");
		}

		// send
		if (send(s, petition, sizeof(petition), 0) == SOCKET_ERROR) {
			errorMessage("Error al enviar una cadena.");
		}

		// receive
		if (recv(s, response, sizeof(response), 0) != RESPONSE_SIZE) {
			errorMessage("Error al recibir la respuesta.");
		}

		cout << ". ";

		// close
		if (closesocket(s) != 0) {
			errorMessage("Error al cerrar el socket.");
		}

		float tiempo = GenerateExponentialDistribution((float)reflexTime);

		threadInfo[threadNum].reflex[i] = tiempo;
		threadInfo[threadNum].contPet++;

		Sleep(tiempo*1000);
	}

	return 0;
}


int main(int argc, char *argv[]) {
	HANDLE handleThread[MAXUSERS];
	int parametro[MAXUSERS];

	if (argc != 4) {
		cout << "Introducir num. usuarios: ";
		cin >> totalUsers;
		cout << "Introducir num. peticiones por usuario: ";
		cin >> petitionsPerUser;
		cout << "Tiempo de reflexion despues de cada peticion: ";
		cin >> reflexTime;
	}
	else {
		totalUsers = atoi(argv[1]);
		petitionsPerUser = atoi(argv[2]);
		reflexTime = atof(argv[3]);

		cout << "Num. usuarios: " << totalUsers << endl;
		cout << "Num. peticiones: " << petitionsPerUser << endl;
		cout << "Tiempo de reflexion: " << reflexTime << endl;
	}

	cout << "Utilizando IP " << SERVERIP << endl;

	if (totalUsers > MAXUSERS || petitionsPerUser > MAXPETITIONS) {
		errorMessage("Arumentos invalidos.");
	}

	// init socket connection
	WORD wVersionRequested = MAKEWORD(2, 0);
	WSAData wsaData;
	if (WSAStartup(wVersionRequested, &wsaData) != 0) {
		errorMessage("Ha ocurrido un error al inicializar el uso de sockets.");
	}

	if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 0) {
		errorMessage("La liberia no soporta la version 2.0");
	}
```

```
	cout << "Transmitiendo ";
	for (int i = 0; i < totalUsers; i++) {
		parametro[i] = i;
		handleThread[i] = CreateThread(NULL, 0, Usuario, &parametro[i], 0, NULL);
		if (handleThread[i] == NULL) {
			errorMessage("Error al lanzar el hilo.");
		}
	}

	for (int i = 0; i < totalUsers; i++) { // el hilo principal espera por sus hijos.
		WaitForSingleObject(handleThread[i], INFINITE);
	}

	WSACleanup();

	ofstream output("output.csv");
	output << "User,Counter,";
	for (int i = 0; i < totalUsers; i++) {
		output << "Time" << i << ",";
	}
	output << "Total" << endl;

	cout << endl << "RESULTADOS:" << endl;
	for (int i = 0; i < totalUsers; i++) {
		auto cont = threadInfo[i].contPet;
		cout << "Usuario: " << i << ", contador: " << cont;
		output << i << "," << cont << ",";

		float totalReflex = 0;
		for (int j = 0; j < petitionsPerUser; j++) {
			auto reflex = threadInfo[i].reflex[j];
			output << reflex << ",";
			totalReflex += reflex;
		}

		cout << ", tiempo total de espera: " << totalReflex;
		cout << " (medio: " << totalReflex / petitionsPerUser << ")" << endl;
		output << totalReflex << endl;
	}

	output.close();
}
```