PL-01	07	Puga Fernández Maldonado Escobedo	Gonzalo Roberto Carlos
Nº PLo	Equipo	Apellidos	Nombre

71.779.257-Y 73208290	UO277906@uniovi.es UO297453@uniovi.es
DNI	e-mail

1	Desarrollo de un inyector de carga	
Nº 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

1. Resultados del inyector con hilos.
2. Preguntas anexo a la práctica 1.
3. Código fuente del inyector.

1. Resultados del inyector con hilos.

Prueba	Concepto	Valor esperado	Valor obtenido
1	Media del tiempo de reflexion	1	1,235539
	Numero de peticiones por hilo	10	10
2	Media del tiempo de reflexion	1	1,00906201
	Numero de peticiones por hilo	100	100
3	Media del tiempo de reflexion	5	6,177695
	Numero de peticiones por hilo	10	10
4	Media del tiempo de reflexion	5	5,045310
	Numero de peticiones por hilo	100	100

2. Preguntas anexo a la práctica 1

• ¿Todas las filas del archivo del registro contador de rendimiento son significativas? ¿Por qué?

No, las primera filas están midiendo el rendimiento cuando el inyector de carga aún no se había iniciado y últimas filas están midiendo cuando el inyector ya había acabado.

• ¿Cuál es la utilización promedio del procesador y explica cómo la has calculado?

13,7616358 %
Se ha calculado haciendo el promedio del % del procesador.

• ¿Cuál es la utilización promedio de la memoria y explica cómo la has calculado?

58,0793922%

La utilización promedio de memoria se calcula con (Memoria total instalada – memoria caché – memoria disponible) /(Memoria total instalada) *100.

• Como el experimento (inyector y servidor) se ejecutan 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.

Como no hay tráfico en la red, el ancho de banda actual es la totalidad de la red. Suponiendo que el valor medio de contador Total de bytes /s fuera de 850000, la utilización de la red sería del 0,085%
Esto se calcula de la siguiente forma:
% Utilización = (Total de bytes /s /Ancho de banda actual)x100
% Utilización = (850000/1000000000)x100 = 0,085

3. Código fuente del inyector

```
#include <windows.h>
#include <iostream>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <fstream>

using namespace std;
//DAR VALORES A ESTAS CONSTANTES
#define MAXPETICIONES 100
#define MAXUSUARIOS 50

/*
->INTRODUCIR VALORES DE FUNCIONAMIENTO
->BIEN AQUI O LEYENDOLOS COMO VALORES POR TECLADO
*/


int numUsuarios;
int numPeticiones;
float tReflex;

// Estructura de almacenamiento

struct datos {
	int contPet;
	float reflex[MAXPETICIONES];
};

datos datoHilo[MAXUSUARIOS];


//---------------------------------------------------------------------
float NumeroAleatorio(float limiteInferior, float limiteSuperior) {
	float num = (float)rand();
	num = num * (limiteSuperior - limiteInferior) / RAND_MAX;
	num += limiteInferior;
	return num;
}

//---------------------------------------------------------------------
float DistribucionExponencial(float media) {
	float numAleatorio = NumeroAleatorio(0, 1);
	while (numAleatorio == 0 || numAleatorio == 1)
		numAleatorio = NumeroAleatorio(0, 1);
	return (-media) * logf(numAleatorio);
}

//---------------------------------------------------------------------
// Funci�n preparada para ser un thread

DWORD WINAPI Usuario(LPVOID parametro) {

	DWORD dwResult = 0;
	int numHilo = *((int*)parametro);
	int i;
	float tiempo;

	srand(101 + numHilo * 7);

	datoHilo[numHilo].contPet = 0;
```

```
		// ... Resto de cosas comunes para cada usuario

		for (i = 0; i < numPeticiones; i++) {
			// PRINTF solo para depuraci�n NUNCA en medici�n
			printf("Peticion %d para el usuario %d\n", i, numHilo);
			// Hacer petici�n cuando se implementen los sockets
			// ----
			// Calcular el tiempo de reflexi�n antes de la siguiente petici�n
			tiempo = DistribucionExponencial((float)tReflex);

			// Guarda los valores de la petici�n
			datoHilo[numHilo].reflex[i] = tiempo;
			datoHilo[numHilo].contPet++;

			// Espera los milisegundos calculados previamente
			Sleep(tiempo * 1000);
		}
		return dwResult;
}


int main(int argc, char* argv[])
{
		printf("Nuevaaaaaaaa\n");
		int i, j;
		HANDLE handleThread[MAXUSUARIOS];
		int parametro[MAXUSUARIOS];

		// Leer por teclado los valores para realizar la prueba o asignarlos. En este caso, los obtiene de los
argumentos.
		numUsuarios = atoi(argv[1]);
		numPeticiones = atoi(argv[2]);
		tReflex = atof(argv[3]);

		// Lanza los hilos
		for (i = 0; i < numUsuarios; i++) {
			parametro[i] = i;
			handleThread[i] = CreateThread(NULL, 0, Usuario, &parametro[i], 0, NULL);
			if (handleThread[i] == NULL) {
				cerr << "Error al lanzar el hilo" << endl;
				exit(EXIT_FAILURE);
			}
		}

		// Hacer que el Thread principal espere por sus hijos
		for (i = 0; i < numUsuarios; i++)
			WaitForSingleObject(handleThread[i], INFINITE);



		//Calcular el tiempo medio de reflexión total, y indivudal.
		float tiempoMediaTotal = 0;
		for (int i = 0; i < numUsuarios; i++) {
			int num_peticiones_hilo = datoHilo[i].contPet;
			float media_reflexion_hilo = 0;
			for (int j = 0; j < datoHilo[i].contPet; j++) {
				for (int z = 0; z < num_peticiones_hilo; z ++) {
					int tiempo_reflexion = datoHilo[i].reflex[z];
					media_reflexion_hilo += tiempo_reflexion;
				}
				media_reflexion_hilo = (float)media_reflexion_hilo / num_peticiones_hilo;
			}
			tiempoMediaTotal += media_reflexion_hilo;
			//printf("La media del hilo %d es %f\n", i, media_reflexion_hilo);
		}
		tiempoMediaTotal =(float) tiempoMediaTotal / numUsuarios;
		printf("La media total es %f\nEl número de peticiones por hilo es %d\n", tiempoMediaTotal, numPeticiones);


		// Recopilar resultados y mostrarlos a pantalla o
		// guardarlos en disco
		FILE* archivo_resultados;

		fopen_s(&archivo_resultados, "resultados.txt", "w");

		fprintf(archivo_resultados, "NumeroUsuario NumeroPeticiones TiempoReflexion\n");
```

```
	for (i = 0; i < numUsuarios; i++) {

		for (j = 0; j < numPeticiones; j++) {

			fprintf(archivo_resultados, "%d ; %d ; %f\n", i, j, datoHilo[i].reflex[j]);

		}

	}

	fclose(archivo_resultados);



	// Recopilar resultados y mostrarlos a pantalla o guardarlos en disco
	for (int i = 0; i < numUsuarios; i++) {
		//Obtener el número de peticiones por hilo
		int cont = datoHilo[i].contPet;
		//printf("Contpet del hilo %d es %d\n", i, cont);
	}

	/*
	->	POR HACER
	*/

	return 0;
}
```