PL3	2	Santos Gómez Iglesias Manzano	Pablo Pelayo
Nº PLo	Equipo	Apellidos	Nombre

71976794-L 32893351-Q	UO290260@uniovi.es UO266600@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 2024-2025

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

Contenido

Asignatura de....1

CONFIGURACIÓN Y EVALUACIÓN DE SISTEMAS....1

Curso 2024-2025....1

Introducción....2

Tabla....2

Código fuente del programa....2

Preguntas del anexo....5

Conclusión....6

Introducción

En esta prueba crearemos un inyector de carga para enviarle a un servidor, a su vez se tomarán medidas respecto a ciertas medidas como el valor de reflexión. En la segunda parte se configura un monitor de rendimiento para capturar datos del equipo.

Tabla

Prueba	Concepto	Valor esperado	Valor obtenido
1	Media del tiempo de reflexión	1	1,3
	Número de peticiones por hilo	10	10
2	Media del tiempo de reflexión	1	1,039970858
	Número de peticiones por hilo	100	100
3	Media del tiempo de reflexión	5	6,5410
	Número de peticiones por hilo	10	10
4	Media del tiempo de reflexión	5	5,199854269
	Número de peticiones por hilo	100	100

Código fuente del programa

```
#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
        #define PUERTO 57000
        #define TAM_PET 1250
        #define TAM_RES 1250

//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);
}

// Función para cargar la librería de sockets
int Ini_sockets(void) {
	WORD wVersionDeseada;
	WSAData wsaData;

	int error;

	wVersionDeseada = MAKEWORD(2, 0);
	if (error = WSAStartup(wVersionDeseada, &wsaData) != 0) {
		return error;
	}

	// Comprobar si la DLL soporta la versión 2.0

	if (LOBYTE(wsaData.wVersion) != 2 || HIBYTE(wsaData.wVersion) != 0) {
		error = 27;
		cerr << "La librer�a no soporta la versi�n 2.0" << endl;
		WSACleanup();
	}
	return error;
}

// Función para descargar la librería de sockets
void Fin_sockets(void) {
	WSACleanup();
}

//-------------------------------------------------------------------------
// Función preparada para ser un thread

DWORD WINAPI Usuario(LPVOID parametro) {

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

	//Variables para los sockets

	SOCKET elSocket;
	sockaddr_in dirServidor;
	char peticion[TAM_PET];
	char respuesta[TAM_RES];
	int valorRetorno;  // Para control de errores


	//ESCOGER LOS VALORES PARA INICIALIZAR LA SEMILLA ALEATORIA

		srand(57 + numHilo * 7);

	elSocket = socket(AF_INET, SOCK_STREAM, 0);
	if (elSocket == INVALID_SOCKET) {
```

```
                //mensaje y salida del programa
                printf("Socket invalido.");
                WSACleanup();
                exit(EXIT_FAILURE);
        }

        // 2 .- Conexión con el servidor
                //

        dirServidor.sin_family = AF_INET;
                dirServidor.sin_addr.s_addr = inet_addr("127.0.0.1");
                dirServidor.sin_port = htons(PUERTO + numHilo);
        valorRetorno = connect(elSocket, (struct sockaddr*)&dirServidor,
sizeof(dirServidor));
        if (valorRetorno == SOCKET_ERROR) {
                //programa aborta
                printf("Error en el socket.");
                closesocket(elSocket);
                WSACleanup();
                exit(EXIT_FAILURE);
        }

        // 3 .- Enviar una cadena
                //
        valorRetorno = send(elSocket, peticion, sizeof(peticion), 0);
        if (valorRetorno == SOCKET_ERROR) {
                printf("Error en el socket al enviar cadena.");
                closesocket(elSocket);
                WSACleanup();
                exit(EXIT_FAILURE);
        }

        // 4 .- Recibir la respuesta
        //
        valorRetorno = recv(elSocket, respuesta, sizeof(respuesta), 0);
        if (valorRetorno != TAM_RES) {
                //Se cierra el programa
                printf("Tamaño de respuesta invalido.");
                closesocket(elSocket);
                WSACleanup();
                exit(EXIT_FAILURE);
        }

        // 5 .- Cerrar la conexión
        //

        closesocket(elSocket);

        // Fin de la petición

        datoHilo[numHilo].contPet = 0;

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

        for (i = 0; i < numPeticiones; i++) {
                // PRINTF solo para depuraci□n NUNCA en medición
                printf("Peticion %d para el usuario %d\n", i, numHilo);
                // Hacer petición cuando se implementen los sockets
                // ----
                // Calcular el tiempo de reflexi□n antes de la siguiente petición
                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[])
{
        int i, j;
        HANDLE handleThread[MAXUSUARIOS];
        int parametro[MAXUSUARIOS];
```

```
	// Leer por teclado los valores para realizar la prueba o asignarlos

	//POR HACER(Usar argc y argv // atoi y/o atof)
	if (argc != 4) {
		printf("USO:numUsuarios peticiones  tiempo de reflexion \n");
		exit(EXIT_FAILURE);
	}
	else {
	numUsuarios = atoi(argv[1]);
	numPeticiones = atoi(argv[2]);
	tReflex = atof(argv[3]);
	FILE* archivo;

	fopen_s(&archivo,"Resultados.txt","w");

	// Inicializar los sockets UNA sola vez en el programa

	Ini_sockets();

		// 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);


		// Descargar la librerIa de sockets, aqui o donde se acabe
	// el programa

		Fin_sockets();

	// Recopilar resultados y mostrarlos a pantalla o
	// guardarlos en disco
		fprintf(archivo,"Usuario ; Numero de petición ; Tiempo de reflexion \n");

		for (int i = 0;i < numUsuarios;i++) {
			for (int j = 0;j < numPeticiones;j++) {
				fprintf(archivo, "%d; %d; %f \n", i, j,
datoHilo[i].reflex[j]);
			}
		}

		fclose(archivo);

}

		return 0;
}
```

Preguntas del anexo

¿Todas las filas del archivo del registro contador de rendimiento serían igual de significativas? ¿Por qué?

No, ya que la primera línea del registro de contador de rendimiento se encuentra vacía debido a que no se ha inicializado el inyector de carga. A su vez si tardamos en desactivar el contador de rendimiento la última línea también sería vacía o contendría datos irrelevantes.

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

El porcentaje de promedio es de 5,3214 %. Para calcular el promedio del procesador simplemente hacemos una media aritmética del porcentaje de tiempo del procesador de los datos recopilados por el monitor.

¿Cuál es la utilización promedio de la memoria y explica cómo la has calculado?
La utilización promedio de la memoria es de 47,827177% y para ello se calcularía

$$\%\text{ de la memoria} = \frac{(\text{memoria total instalada (bytes)} - \text{memoria caché} - \text{memoria disponible})}{(\text{memoria total instalada})}$$

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.

El ancho de banda actual es de 1000000000 viene expresado en bits/s y en su totalidad ya que no hay tráfico en la red.

Para calcular el %utilización de red

$$\%\text{utilización de red} = \frac{\text{Total de byte/s}}{\text{Ancho de banda actual byte/s}} \times 100 \rightarrow \frac{850000}{125000000} \times 100 = 0{,}68\%$$

Conclusión

Con este ejercicio aprendemos a hacer un inyector de carga y a medir el rendimiento de un sistema mediante el uso de varios hilos, a simular peticiones a un servidor y con el monitor a tomar captura de datos de rendimiento del sistema. Gracias al monitor podemos tener una visión mucho más clara de la utilización de los recursos del sistema al iniciar el inyector de carga.