PRÁCTICA 1. MPI

1. INTRODUCCIÓN

En este documento se describe la práctica 1 de la asignatura Computación de Altas Prestaciones del Grado de Ingeniería de Computadores.

Esta práctica es la primera de las dos prácticas obligatorias de la asignatura.

2. DESCRIPCIÓN GENERAL

Esta práctica tiene como objetivo poner en práctica los conocimientos adquiridos sobre MPI durante el transcurso de la asignatura.

Para ello será necesario modificar un programa dado para mejorarlo mediante la aplicación de estrategias basadas en las capacidades de MPI. El programa descifra mediante fuerza bruta un texto encriptado mediante un algoritmo parecido al empleado por la máquina enigma de la segunda guerra mundial.

Se recomienda una lectura detallada del enunciado completo para tener una visión global del problema a resolver y no pasar por alto ninguno de los detalles descritos.

3. DESCIFRANDO ENIGMA

Aunque el programa proporcionado junto con este enunciado ya implementa y resuelve la descodificación del mensaje de entrada, es necesario conocer el algoritmo que emplea para poder aplicar de la mejor forma posible las técnicas aprendidas de MPI con el objetivo de optimizar la eficiencia del programa. En los siguientes apartados, por tanto, se describirá este mecanismo de cifrado que está basado en la máquina enigma [3].

CIFRADO POR DESPLAZAMIENTO

El cifrado por desplazamiento o cifrado César [1] es una de las técnicas de cifrado más sencillas, antiguas y empleadas. Consiste en que cada letra del texto original se sustituye por otra que se encuentra desplazada un número fijo de posiciones en el alfabeto utilizado. Por ejemplo, utilizando el abecedario propio del castellano y un desplazamiento de valor 5, la letra A se convertiría en la letra F, la B en la G, la C en la H, etc.

EL CÓDIGO ASCII

El alfabeto utilizado por el programa no corresponde al castellano, sino que el sistema utiliza el código ASCII [2], que es un código de caracteres basado en el alfabeto latino (en su versión básica no está incluida la letra Ñ) y funciona asignando un valor numérico a cada carácter.

El primer grupo de caracteres se denomina grupo de caracteres de control y en este se incluyen caracteres no imprimibles que representan, por ejemplo, retornos de línea, tabulaciones u otras instrucciones especiales de control de periféricos.

El segundo grupo de caracteres se denomina grupo de caracteres imprimibles y corresponden a los valores desde el 32 (espacio) hasta el 127 (suprimir). Entre ellos podemos encontrar el 0 (valor 48), la letra A mayúscula (valor 65) o la a minúscula (valor 97). En la Tabla 1 se muestran algunos de estos caracteres imprimibles.

Tabla 1. Fragmento del conjunto de caracteres imprimibles del Código ASCII

Valor	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
Carácter	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p

Dado que este alfabeto no incluye los caracteres especiales del castellano como tildes y eñes los textos cifrados para los diferentes casos de prueba son en inglés.

De este modo, por ejemplo, la palabra "enigma" correspondería a los valores: 101, 110, 105, 103, 109, 97. Si sobre estos valores se aplica un desplazamiento de 2, la palabra resultante vendría dada por los valores: 103, 112, 107, 105, 111, 100, que daría como resultado: "gpkiod".

Lo que facilita el uso de este código como alfabeto en el sistema de cifrado es que una variable de tipo *char* se convierte a una variable de tipo entero empleando esta codificación, por ejemplo: la variable de c de tipo *char* cuyo valor es 'a' se convierte en una variable de tipo *int* de valor 97. Debido a esto es muy sencillo implementar mecanismos de cifrado de texto por sustitución jugando con conversiones de variables de tipo *char* a *int* y viceversa.

ROTORES

La máquina enigma codificaba el texto de entrada mediante desplazamiento, pero en lugar de realizar este desplazamiento una única vez, realizaba el proceso encadenando salidas y entradas para cada uno de los rotores mecánicos que constituían la máquina. Es decir, el primer rotor cifraba por desplazamiento un carácter del texto de entrada, ese carácter cifrado se convertía en la entrada del segundo rotor que volvía a aplicar desplazamiento y la salida de este se convertía en la entrada del tercer rotor que aplicaba el desplazamiento de nuevo.

Además, para complicar algo más el proceso, el desplazamiento que aplicaba cada rotor no era fijo, sino que se incrementaba con cada operación. Una configuración típica podía ser que el primer rotor incrementaba en uno el desplazamiento cada vez que cifraba un carácter, cuando el rotor daba una vuelta completa (26 posiciones) entonces era el segundo rotor el que incrementaba su desplazamiento y cuando este completaba una vuelta, se incrementaba el desplazamiento del tercer rotor. Este mecanismo de desplazamientos variables producía que dos caracteres iguales en la secuencia de entrada, por ejemplo: AA, resultaban dos caracteres diferentes en el texto cifrado final, por ejemplo: QR.

La clave, por tanto, para saber descifrar un mensaje encriptado con la máquina enigma consistía en conocer la disposición inicial de los rotores, ya que esto definía cómo sería el desplazamiento aplicado y la variación de este.

IMPLEMENTACIÓN PARA LA PRÁCTICA

La adaptación del mecanismo de la máquina enigma para esta práctica resulta en el siguiente algoritmo:

- El cifrado se realiza mediante operaciones de desplazamiento de caracteres según el código ASCII.
- Las operaciones de desplazamiento se encadenan en varias fases (equivalente al número de rotores en la máquina enigma). Cada etapa cifra el texto completo y el texto completo cifrado es la entrada de la siguiente etapa.

- Cada fase de desplazamiento está caracterizada por un valor inicial de desplazamiento.
- Cada fase de desplazamiento incrementa en uno el valor del desplazamiento aplicado con cada carácter procesado.
- La clave que define el cifrado es un número de n cifras siendo n el número de rotores o de etapas de cifrado por desplazamiento.
- El valor inicial de cifrado de una etapa está comprendido entre 0 y 9 (para la primera etapa se asume que como mínimo tendrá un valor de 1).

Por ejemplo, el cifrado de la palabra "enigma" con un sistema de dos rotores con la clave 35 sería del siguiente modo:

1) Convertir enigma a valores numéricos según el código ASCII: 101, 110, 105, 103, 109, 97
2) Cifrar el primer carácter por desplazamiento mediante el valor 3: 104
3) Cifrar el segundo carácter por desplazamiento mediante el valor 4: 114
4) Cifrar el tercer carácter por desplazamiento mediante el valor 5: 110
5) Cifrar el cuarto carácter por desplazamiento mediante el valor 6: 109
6) Cifrar el quinto carácter por desplazamiento mediante el valor 7: 116
7) Cifrar el sexto carácter por desplazamiento mediante el valor 8: 105
8) El resultado de la primera etapa de cifrado es: 104, 114, 110, 109, 116, 105, que corresponde al texto: "hrnmti".
9) Se aplica la segunda etapa de cifrado empezando por un desplazamiento de 5.
10) Cifrar el primer carácter por desplazamiento mediante el valor 5: 109
11) Cifrar el segundo carácter por desplazamiento mediante el valor 6: 120
12) Cifrar el tercer carácter por desplazamiento mediante el valor 7: 117
13) Cifrar el cuarto carácter por desplazamiento mediante el valor 8: 117
14) Cifrar el quinto carácter por desplazamiento mediante el valor 9: 125
15) Cifrar el sexto carácter por desplazamiento mediante el valor 10: 115
16) El resultado del cifrado es: 109, 120, 117, 117, 125, 115, que corresponde a: "mxuu}s".

De este modo, a un sistema de n etapas le corresponde una clave de n cifras, por ejemplo: para 3 rotores las claves posibles irían desde el 100 hasta el 999 (el primer rotor debe tener como mínimo un desplazamiento inicial de 1); para 4 etapas desde el 1000 hasta el 9999, etc. Siendo cada una de las cifras de la clave el valor de desplazamiento inicial de cada rotor, por ejemplo, la clave 2579 implicaría un desplazamiento inicial de valor 2 para la primera etapa, un desplazamiento inicial de valor 5 para la segunda etapa, de valor 7 para la tercera y un desplazamiento inicial de 9 para la cuarta y última fase de cifrado.

El mecanismo para descifrar consiste en aplicar el desplazamiento en el sentido contrario, restando, y contemplando además que el valor de desplazamiento de cada rotor se reduce en una unidad con cada carácter procesado.

4. CÓDIGO INICIAL

El código proporcionado realiza un proceso de descifrado del texto de entrada mediante fuerza bruta, es decir va probando el algoritmo de descifrado con todas las claves posibles para el número de rotores dados. Para comprobar que el texto se ha descifrado correctamente, cada línea del texto de entrada comienza con la clave utilizada para cifrarla.

Por ejemplo, en el caso de prueba proporcionado con el programa, el texto original de entrada es el siguiente:

```
I am the watcher on the walls. I am the fire that burns against the cold, the light that brings the dawn, the horn that wakes the sleepers, the shield that guards the realms of men. I pledge my life and honor to the Night's Watch, for this night and all the nights to come.
```

Sin embargo, el texto que finalmente se cifra consiste en el texto original dividido en líneas del mismo número de caracteres e incluyendo al principio de cada línea el valor de la clave usada para cifrar esa línea:

```
73-I am the watcher on the walls.
17-I am the fire that burns again
13-st the cold, the light that br
29-ings the dawn, the horn that w
23-akes the sleepers, the shield
61-that guards the realms of men.
17-I pledge my life and honor to
83-the Night's Watch, for this ni
41-ght and all the nights to come
```

Esto corresponde con el array de datos enteros que se muestra en Código 1.

```
int ciphered_keys2[9][33] = {
 {65,63,59,89,50,117,131,56,142,132,131,64,153,133,154,139,146,145,160,80,161,162,86,172,162,161,94,183,163,176,178,187,120},
 {57,65,57,87,48,115,129,54,140,130,129,62,134,139,150,139,72,158,148,143,164,82,150,171,170,168,175,94,161,169,165,175,182},
 {53,57,53,125,128,46,132,122,121,54,123,137,136,130,76,66,152,142,141,74,152,151,151,154,168,86,172,162,157,178,96,164,182},
 {61,70,60,122,129,124,138,57,143,133,132,65,135,134,158,151,87,77,163,153,152,85,159,168,173,171,95,181,171,166,187,105,194},
 {55,58,54,108,120,116,132,51,137,127,126,59,144,139,134,136,149,140,155,158,89,79,165,155,154,87,172,163,166,164,173,167,101},
 {61,58,56,129,119,114,135,53,126,142,124,143,131,148,67,153,143,142,75,159,148,146,159,162,170,89,170,163,95,174,168,179,117},
 {57,65,57,87,48,130,128,123,124,129,129,62,141,155,68,146,145,144,145,78,145,160,152,86,160,169,170,173,178,98,184,181,104},
 {67,64,60,133,123,122,55,103,132,132,135,149,8252,152,71,128,140,161,146,153,95,85,157,168,173,93,179,169,172,184,103,183,180},
 {57,56,54,114,117,131,49,116,131,123,57,124,137,139,65,151,141,140,73,153,150,150,153,167,168,87,173,170,93,162,176,176,170},
};
```

Código 1. Array de datos para la entrada cifrada del ejemplo incluido en el código proporcionado.

Que al imprimirlo por pantalla genera lo siguiente:

```
A?;Y2uâ8Äää@ÖàÜïÆæáPíóV¼óí^Àú▒▓╗x
9A9W0sü6îéü>åïûïH×öÅñRû½¬¿»^í®Ñ»Â
595}Ç.äzy6{ëêéLBÿÄìJÿùùÜ¿V¼óØ▓`ñÂ
=F<zü|è9ÅàäAçå×ùWMúÖÿUf¿¡½_Á½º╗i┬
7:6lxtä3ëº~;Éïåêòîø×YOÑøÜW¼úªñ¡ºe
=:8üwrç5~Ä|ÅâöCÖÅÄKƒöÆfó¬Y¬ú_«¿|u
9A9W0éÇ{|üü>ìøDÆæÉæNæáÿVá®¬¡▓b©Áh
C@<à{z7gääçò<ÿGÇîíÆÖ_UØ¿¡]|®¼©gÀ┤
```

```
986ruâ1tâ{9|ëïAùìîIÖûûÖº¿W¡¬]ó¬
```

Al ejecutar el programa proporcionado se resuelve el descifrado de este texto y se obtiene el original. El programa no aplica ninguna técnica de MPI y consigue resolver el problema en milisegundos, pero ¿qué ocurre si se pretende descifrar un texto de más líneas, de más caracteres por línea y, sobre todo, empleando claves de cifrado de varias cifras (lo que implica varias etapas de cifrado)? Para poder responder a esta pregunta, se proporcionan también los *arrays* de datos correspondientes a textos cifrados con el mismo mecanismo, pero de mayor extensión en las tres dimensiones citadas (número de líneas, longitud de líneas y tamaño de las claves de cifrado).

Se pueden resolver estos textos cifrados usando el mismo programa, pero el objetivo de la práctica es intentar resolver el descifrado de estos textos empleando las técnicas de MPI desarrolladas durante la asignatura.

Loas funciones implementadas en el programa son tres: *printNumbersAsString* (ver Código 2), *decipher* (Código 3) y *enigma* (Código 4).

```
void printNumbersAsString(int lines[nLines][nCharsPerLine])
{
    for (int idx = 0; idx < nLines; idx++)
    {
        char line[nCharsPerLine+1];
        for (int idx2 = 0; idx2 < nCharsPerLine; idx2++)
        {
            line[idx2] = lines[idx][idx2];
        }
        line[nCharsPerLine] = '\0';
        printf("%s\n", line);
    }
}
```

Código 2. Función que permite imprimir por consola el array de enteros conforme a la codificación ASCII.

La función *printNumbersAsString* convierte la matriz de números enteros que recibe como argumento y en la que se encuentra codificado un texto en cadenas de caracteres de C que imprime por consola. Esta función permite pasar de los valores enteros con los que trabaja el mecanismo de cifrado y descifrado a caracteres de texto imprimibles por pantalla. Se puede emplear tanto para el texto cifrado como para el descifrado finalmente.

```
int* decipher(int line[], int key)
{
    int rawData[nCharsPerLine];

    for (int idx = 0; idx < nCharsPerLine; idx++)
    {
        rawData[idx] = line[idx];
    }

    int* rotorKeys = (int*)malloc(sizeof(int) * nRotors);
```

```
        int remainder = key;
        for (int idx = 0; idx < nRotors; idx++)
        {
            int divisor = pow(10, (nRotors - (1 + idx)));
            rotorKeys[idx] = (int)(remainder / divisor);
            remainder = (int)(remainder % divisor);
        }


        for (int rotorIdx = 0; rotorIdx < nRotors; rotorIdx++)
        {
            int displacement = rotorKeys[rotorIdx];
            for (int idx = 0; idx < nCharsPerLine; idx++)
            {
                rawData[idx] = rawData[idx] - displacement++;
            }
        }

        free(rotorKeys);
        return rawData;
    }
```

Código 3. Función que decodifica un array de número enteros con la clave proporcionada.

La función *decipher* realiza el proceso de descifrado conforme al algoritmo de enigma. Para ello somete al proceso de descifrado al array de datos enteros recibido como argumento conforme a la clave recibida igualmente como argumento. El número de cifras de la clave determina automáticamente el número de rotores o etapas de cifrado del mecanismo.

```
    void enigma()
    {
        printf("ESTO ES LA ENTRADA: \n");
        printNumbersAsString(ciphered_keys2);
        printf("\n");
        printf("\n");

        int deciphered[nLines][nCharsPerLine];
        for (int idx = 0; idx < nLines; idx++)
        {
            for (int lineKey = (int)pow(10, nRotors - 1); lineKey < (int)pow(10,
nRotors); lineKey++)
            {
                int* p_deciphered = decipher(ciphered_keys2[idx], lineKey);

                char decipheredLine[nCharsPerLine];
                for (int idx = 0; idx < nCharsPerLine; idx++)
```

```
            {
                decipheredLine[idx] = p_deciphered[idx];
            }

            char stringKey[nRotors + 1];
            sprintf_s(stringKey, "%d", lineKey);
            if (!strncmp(stringKey, decipheredLine, nRotors))
            {
                for (int idx2 = 0; idx2 < nCharsPerLine; idx2++)
                {
                    deciphered[idx][idx2] = decipheredLine[idx2];
                }
                printf("Descifrada linea para %d\n", lineKey);
                break;
            }
        }
    }

    printf("\n");
    printf("ESTO ES LA SALIDA:\n");
    printNumbersAsString(deciphered);
    printf("\n");
    printf("\n");

}
```

Código 4. Función principal que descifra por fuerza bruta el texto de entrada codificado.

La función enigma contiene el algoritmo general del programa que se encarga de, por cada línea del texto de entrada, probar una decodificación con cada una de las claves posibles y comprobar si el resultado es el correcto o no (para ello compara los primeros caracteres de la línea decodificada con la clave con la que se ha intentado descifrar, en caso de que coincida el descifrado se ha realizado con éxito y se agrega esta línea al texto de salida final).

5. OBJETIVO DE LA PRÁCTICA

Tal y como se ha ido comentando en apartados anteriores, lo que se pide es mejorar el programa proporcionado aplicando técnicas de MPI. Será necesario justificar las decisiones de diseño llevadas a cabo sobre todo en términos de la estrategia empleada.

El programa original puede ser modificado con libertad, pero se recomienda no perder de vista el caso básico de ejemplo y comprobar que tras cada modificación el programa sigue descifrando correctamente el texto de muestra. Una vez se haya comprobado que las modificaciones son válidas, se puede aplicar el programa a los casos más complejos y analizar los resultados.

Como primera aproximación se puede plantear dividir el problema en diferentes tareas, por ejemplo, una tarea por cada línea del código original. Dentro de esa tarea habrá que probar todas las posibles claves de cifrado hasta

encontrar la clave correcta. En última instancia es necesario combinar los resultados obtenidos para mostrar por consola el texto completo descifrado.

Para enriquecer el análisis de las posibles soluciones se pueden incluir instrucciones en el código para medir tiempos totales de resolución del problema y así comparar las diferentes estrategias evaluadas.

En la memoria será necesario razonar los resultados obtenidos y exponer las ventajas de la estrategia MPI implementada. Dentro de este análisis pueden responderse las siguientes preguntas:

- ¿Hay mejora considerable en cuanto al tiempo de ejecución usando MPI o sin usarlo? ¿Por qué?
- ¿Tiene sentido utilizar un esquema maestro/esclavo? ¿Por qué?
- ¿Cómo se utiliza la comunicación entre procesos en la solución?

6. CRITERIOS DE EVALUACIÓN

Aparte de los requisitos descritos, se valorará la claridad, limpieza, organización y planteamiento de la memoria: se tendrá en cuenta la calidad de esta (ver apartado 9), ya que la memoria representa el resultado del trabajo realizado. Como consecuencia, esta práctica podrá ser considerada suspensa si la memoria no alcanza un nivel mínimo de calidad.

7. METODOLOGÍA

La práctica se realiza en grupo y se entrega a través del campus virtual antes de la fecha límite fijada para ello.

Al igual que cualquier otro trabajo, se aplicará, llegado el caso, la normativa anticopia de la universidad.

8. ENTREGABLES

Se deben entregar, por cada grupo, dos archivos comprimidos (zip o 7z):

- Memoria de la práctica (*pdf)*, debe seguir las recomendaciones descritas en el apartado 9.
- Archivo comprimido que incluya el código desarrollado.

9. RECOMENDACIONES PARA LA REDACCIÓN DE MEMORIAS DE PRÁCTICAS

Se recomienda seguir las recomendaciones de escritura de documentos técnicos en general y de memorias de trabajos prácticos en particular. Principalmente:

- Las memorias se entregan en formato *pdf*.
- El documento debe incluir:
    - Portada: con información sobre curso, grado, asignatura, autores y título.
    - Índice. Opcionalmente, índice de tablas, de figuras y de expresiones matemáticas.
    - Introducción: con una descripción breve de la finalidad del documento, contexto y contenido de la memoria.
    - Conclusiones: consideraciones sobre lo que ha supuesto la realización del trabajo.
    - Bibliografía: listado de referencias empleadas en el documento (usar referencias cruzadas para referirse a ellas en el documento).
- Usar un lenguaje basado en formas no personales.

- Incluir siempre títulos en las figuras, tablas y expresiones matemáticas (si se diera el caso), de forma que se pueda hacer referencia a ellas en cualquier parte del texto.
- Incluir paginación en el documento.
- Numerar los apartados, capítulos o secciones para facilitar la referencia a ellos si fuera necesario.
- Si se incluye código en la memoria, no incluirlo como imagen sino como texto aplicando un formato distinto.
- Revisar a conciencia la ortografía y gramática, incluyendo los signos de puntuación.
- Justificar el texto.
- Si el documento incluye gráficas, estas deben incluir títulos y unidades de cada eje y una leyenda con la descripción de cada serie de datos. También título y opcionalmente un subtítulo.
- Se deben minimizar los espacios en blanco entre figuras o tablas, el texto debe rellenarlos y hacer referencia a las figuras o tablas cuando corresponda.

REFERENCIAS

[1] https://es.wikipedia.org/wiki/Cifrado_C%C3%A9sar
[2] https://es.wikipedia.org/wiki/ASCII
[3] https://es.wikipedia.org/wiki/Enigma_(m%C3%A1quina)