NIP:841886
Autor: Rubén Martín González
NIP:811019
Autor: Sergio Salesa Quintanilla

Descripción de los datos compartidos:

- Las variables N_EST,N_FIL y N_COL son constantes de tipo entero que definen el número de estudiantes , el número de filas y columnas que tendrá la correspondiente matriz.
- Una matriz D de tipo entero con N_FIL filas y N_COL columnas que guardará los elementos a leer.
- Crearemos una variable entera “libre” que guardará el número de sillas libres, y un vector de enteros “examinado” con dos posiciones donde guardaremos la posición en el vector nips del alumno sentado.
- Vamos a crear también dos vectores de tipo entero con N_EST componentes, asigFil y parejas donde en el primero asignaremos la fila correspondientes a cada nip y en el segundo guardaremos todos los nips de la correspondiente pareja.
- Necesitaremos un vector de booleanos llamado fin in de N_EST componentes para comunicar cuando el nip menor calcula su parte.
- Y por último necesitaremos un vector de enteros result de N_EST componentes para guardar el resultado de la pareja con nip menor.
- Creamos una variable entera terminados inicializada a 0 donde sumaremos uno por cada pareja finalizada.

Procesos que los comparten:

Comparten: N_EST procesos Estudiante y un proceso Profesor.

Explicación de las restricciones de sincronización a resolver:

Hay que resolver que solamente realizan cada examen dos estudiantes.
También, se tiene que resolver que hasta que el profesor no pase el nip de la pareja y la fila en la que se trabaja al estudiante, este permanezca en espera sin ejecutarse.
Además, hay que gestionar la sincronización entre los dos estudiantes que realizan el examen, para que el de mayor nip espere a que el de menor nip haya dado su resultado correspondiente para mostrar ambos.

Diseño:

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
const int N_EST = 60; // # de estudiantes
const int N_FIL = N_EST /2; // # de filas en la matriz
const int N_COL = 1000; // # de columnas
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
void meterDatos(int D[][N_COL],string nombre){
	ifstream f(nombre);
	if (f.is_open())
	{
		for (unsigned int i = 0; i < N_FIL; i++)
		{
			for (unsigned int j = 0; j < N_COL; j++)
			{
				f >> D[i][j];
			}
		}
	}
}
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
/// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
// Pre : < fila > es un índice de fila de <D >
// Post : devuelve el m´a ximo de la fila < fila >
int maxFila ( int D [ N_FIL ][ N_COL ] , int fila ) {
	int max = 0;
	for (unsigned int i = 0; i < N_COL; i++)
	{
		if (D[fila][i]>max)
		{
		max=D[fila][i];
		}
	}
	return max ;
}

// Pre : < fila > es un´ı ndice de fila de <D >
// Post : devuelve la suma de los els . de la fila < fila >
int sumaFila ( int D [ N_FIL ][ N_COL ] , int fila ) {
	int sum = 0;
	for (unsigned int i = 0; i < N_COL; i++)
```

```
        {
                sum = sum + D[fila][i];
        }
        return sum ;
}
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
void Estudiante (int nip,int& libre,int examinado[2],int parejas[N_EST],int D [ N_FIL ][
N_COL
],bool fin[N_EST],int result[N_EST] ,int& terminados) {
        <await libre>0
                libre--;
                examinado[libre]=nip;
        >
        <await parejas[nip]!=-1> // esperar me sea asignada pareja y fila
        if (nip < miPareja) {
                res[parejas[nip]]=maxFila(D,fila); // calcular max de mi fila
                <fin[pareja[nip]]=true;> // hacerselo llegar a mi pareja
        }
        else {
                int suma=sumaFila(D,fila); // calcular la suma de mi fila
        <await fin[nip]>
        cout<<"El máximo de la fila es "+to_string(res[nip])+" y la suma de la fila es
        "+to_string(suma)<<endl; // coger info de max ( de mi pareja )
        // mostrar resultados
        <terminados++> // comunicar finalización
        }
}
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
void Profesor (int& libre,bool fin[N_EST], int parejas[N_EST],int examinado[2],int
        asigFila[N_FIL],int& filaExam) {
        for ( int i =0; i < N_FIL ; i ++) {
                <await libre===0
                        pareja[examinado[1]]=examinado[0];
                        pareja[examinado[0]]=examinado[1];
                        asigFila[examinado[1]]=i;
                        asigFila[examinado[0]]=i;
                        libre=2;
                >
                // esperar a que haya dos
                // comunicar a cada uno su pareja , y la fila que les toca
        }
        <await filaExam==N_FIL>
        // esperar que todos hayan terminado
}
```

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
int main () {
        int D [ N_FIL ][ N_COL ]; // para almacenar los datos
        int filaExam = 0; // filas completadas
        int parejas [ N_EST ]; // pareja [ i ] ser´a la pareja asignada
        int examinado[2];
        int result[N_EST];
        bool fin[N_EST]={false};
        int libre=2;
        string nom = "datos.txt";
        meterDatos(D,nom);          // cargar " datos . txt " en " D "

        thread th_1(&Profesor,ref(libre),fin,parejas,examinado,asigFila,ref(filaExam));
        //se inician los procesos
        for(int i=0;i<N_EST;i++){
                P[i]=thread(&Estudiante,i,ref(libre),examinado,parejas,D,fin
                ,result,ref(filaExam));
        }
        th_1.join(); //se bloquean los procesos
        for(int i=0;i<N_EST;i++){
                P[i].join();
        }
        cout << " Prueba finalizada \n " ;
        return 0;
}
```