

# Programación 1

## Tema 16

---

Ficheros: otras posibilidades



Escuela de  
Ingeniería y Arquitectura  
**Universidad** Zaragoza





# Objetivos

---

- Trabajo de forma no secuencial con ficheros
  - Modo *append*
  - Acceso directo
  - Modo entrada y salida



# Objetivos

---

- Trabajo de forma no secuencial con ficheros
  - **Modo *append***
  - Acceso directo
  - Modo entrada y salida

# Añadir datos a un fichero

```
/*  
 * Pre: «nombreFichero» hace referencia a un  
 * fichero de texto existente y  
 * modificable.  
 * Post: Inserta al final del fichero de  
 * denominado «nombreFichero» una línea  
 * completa cuyo contenido sea la  
 * cadena de caracteres «linea».  
 */  
void unaLineaAdicional(  
    const char nombreFichero[],  
    const char linea[]);
```

# Una solución

```
void unaLineaAdicional(const char nombreFichero[],
                      const char linea[]) {
    const char FICHERO_TEMPORAL[] = "temporal.tmp";
    ifstream fOriginal;
    fOriginal.open(nombreFichero);
    if (fOriginal.is_open()) {
        ofstream fTemporal;
        fTemporal.open(FICHERO_TEMPORAL);
        if (fTemporal.is_open()) {
            ...
        }
        else {cerr << "No se ha podido escribir el fi..." << endl;
              fOriginal.close(); }
    }
    else { cerr << "No se ha podido leer el fichero ..." << endl;}
}
```

# Una solución

```
void unaLineaAdicional(const char nombreFichero[],  
                       const char linea[]) {
```

```
    ...
```

```
    char c;  
    fOriginal.get(c);  
    while (!fOriginal.eof()) {  
        fTemporal.put(c);  
        fOriginal.get(c);  
    }  
    fTemporal << linea << endl;  
    fTemporal.close();  
    fOriginal.close();  
    remove(nombreFichero);  
    rename(FICHERO_TEMPORAL, nombreFichero);
```

```
    ...
```

```
}
```

Funciones remove y rename  
definidas en <cstdio>

# Función rename

- Biblioteca <stdio>
- **int rename ( const char oldname[], const char newname[] );**
  - **Rename file**
  - Changes the name of the file or directory specified by *oldname* to *newname*.
  - This is an operation performed directly on a file; No streams are involved in the operation.
  - If *oldname* and *newname* specify different paths and this is supported by the system, the file is moved to the new location.
  - If *newname* names an existing file, the function may either fail or override the existing file, depending on the specific system and library implementation.
  - Proper file access shall be available.
- **Parameters**
  - *oldname*: C string containing the name of an existing file to be renamed and/or moved. Its value shall follow the file name specifications of the running environment and can include a path (if supported by the system).
  - *newname*: C string containing the new name for the file. [...]
- **Return value**
  - If the file is successfully renamed, a zero value is returned.
  - On failure, a nonzero value is returned.

# Función remove

---

- Biblioteca <stdio>
- **int remove ( const char filename[] );**
  - **Remove file**
  - Deletes the file whose name is specified in *filename*.
  - This is an operation performed directly on a file identified by its *filename*; No streams are involved in the operation.
  - Proper file access shall be available.
- **Parameters**
  - **filename**: C string containing the name of the file to be deleted. Its value shall follow the file name specifications of the running environment and can include a path (if supported by the system).
- **Return value**
  - If the file is successfully deleted, a zero value is returned.
  - On failure, a nonzero value is returned.



# Una solución mejor

```
void unaLineaAdicional(const char nombreFichero[],
                       const char linea[]) {
    ofstream f;
    f.open(nombreFichero, ios::app);
    if (f.is_open()) {
        f << linea << endl;
        f.close();
    }
    else {
        cerr << "No se ha podido escribir en el fichero"
              << "\"" << nombreFichero << "\"." << endl;
    }
}
```



# Objetivos

---

- Trabajo de forma no secuencial con ficheros
  - Modo *append*
  - **Acceso directo**
  - Modo entrada y salida

# Acceso directo a los datos de un fichero

```
/*  
 * Pre: El fichero de nombre «nombreFichero»  
 * almacena al menos los primeros «i»  
 * números primos, almacenados en orden  
 * ascendente.  
 * Post: Ha devuelto el «i»-ésimo (comenzando a  
 * contar por 1) número primo, según el  
 * contenido del fichero. Si no se ha podido  
 * abrir el fichero, ha escrito un mensaje de  
 * error en «cerr» y ha devuelto un valor  
 * negativo.  
 */  
int leerUnPrimo(const char nombreFichero[], int i);
```

# Una solución

```
int leerUnPrimo(const char nombreFichero[], int i) {  
    ifstream f(nombreFichero, ios::binary);  
    if (f.is_open()) {  
        int primo;  
        for(int j = 1; j <= i; j++) {  
            f.read(reinterpret_cast<char*>(&primo),  
                sizeof(primo));  
        }  
        f.close();  
        return primo;  
    }  
    else {  
        cerr << "No se ha podido leer el fichero \""  
            << nombreFichero << "\"\" << endl;  
        return -1;  
    }  
}
```

# Una solución mejor

```
int leerUnPrimo(const char nombreFichero[], int i) {  
    ifstream f(nombreFichero, ios::binary);  
    if (f.is open()) {  
        f.seekg((i - 1) * sizeof(int));  
        int primo;  
        f.read(reinterpret_cast<char*>(&primo),  
                sizeof(primo));  
        f.close();  
        return primo;  
    }  
    else {  
        cerr << "No se ha podido leer el fichero \""  
                << nombreFichero << "\"" << endl;  
        return -1;  
    }  
}
```

# Objetivos

---

- Trabajo de forma no secuencial con ficheros
  - Modo *append*
  - Acceso directo
  - **Modo entrada y salida**



# Lectura y escritura de datos de un mismo fichero

```
/*  
 * Pre: El fichero de nombre «nombreFichero» contiene los  
 *       primeros números primos, almacenados en orden  
 *       ascendente. El fichero contiene al menos dos primos.  
 * Post: Ha añadido al fichero el número primo que sigue al  
 *        último que tenía inicialmente almacenado. Si no ha  
 *        podido, ha escrito un mensaje de error en «cerr».  
 */  
void agregarSiguientePrimo(const char nombreFichero[]);
```

# Lectura y escritura de datos de un mismo fichero

```
void agregarSiguientePrimo(const char nombreFichero[]) {  
    fstream f(nombreFichero, ios::binary | ios::in | ios::out );  
    if (f.is_open()) {  
        int primo;  
        f.seekg(-1 * sizeof(int), ios_base::end);  
        f.read(reinterpret_cast<char*>(&primo), sizeof(primo));  
  
        primo += 2;  
        while (!esPrimo(primo)) {  
            primo += 2;  
        }  
  
        f.seekp(0, ios_base::end);  
        f.write(reinterpret_cast<const char*>(&primo),  
                sizeof(primo));  
        f.close();  
    } else { ... }  
}
```