CI-3825: Sistemas de Operación I

Proyecto 2 Enero – Marzo 2023

Pokémon FameChecker

Estudiantes:

- Henry Galue (14-10373)

- César Gonzalez (14-10423)

- Eros Cedeño (16-10216)

- Blanyer Vielma (16-11238)

Introducción

El presente informe describe el diseño e implementación de un programa llamado **fame checker,** basado en el objeto clave utilizados en los juegos de la saga **Pokémon.** En este se hacen consultas de los archivos dados como parámetros para dar una respuesta referente al número de entrenadores, región y número de pokémon.

El proyecto contiene una estructura simple y bien documentada, un programa funcional según todos los criterios mencionados en la hoja de requerimientos, y el funcionamiento completamente descrito en este documento. Para el correcto funcionamiento del proyecto son necesarios varios archivos que contengan la información requisito.

Además, el presente informe presentará de manera progresiva el proceso de desarrollo del proyecto, las decisiones de diseño, implementación y descripción de nuestro desarrollo, así como las dificultades presentadas, los métodos utilizados para su desarrollo, finalizando con nuestras conclusiones acerca del resultado final, la cual estará segmentada en dificultades, lecciones aprendidas y consideraciones finales acerca del proyecto.

Decisiones de diseño e implementación

En cuanto a las decisiones sobre la implementación; dividimos los roles de desarrollo para que cada miembro del equipo pudiese ser partícipe del proyecto.

- Con respecto a la arquitectura del proyecto, esta vez debido a la poca cantidad de código implementado, se decidió organizar el proyecto de manera simple y clara. Centralizamos el núcleo del programa en el archivo principal **main.c**, dejando aparte el código que realmente se iba a reutilizar, que en este caso sería el correspondiente a las **estructuras**, que se añadieron en archivos de encabezado aparte.
- Adicionalmente, añadimos un archivo **makefile**, que nos permite construir de manera automatizada el programa de **fameChecker**.

¿Cómo ejecutar el programa?

Para compilar el proyecto basta con correr el comando make en la consola. Posteriormente se generará el archivo **fameChecker** que es el que se usará para la ejecución del proyecto. Para ejecutarlo, escriba el siguiente comando:

```
./fameChecker [-r <region>] [-s <species>] [-t <type>] [-c|--nocount]
[-l|--list] [--size] [name]
```

Llamadas de ejemplo:

Suponga que queremos saber cuántos pokémon pertenecientes a Ash existen que sólo aparecen una vez en todas las regiones. En este caso, podemos hacer la búsqueda:

```
$ ./fameChecker -s pokemon -t one_time “Ash's”
```

Si queremos saber cuáles son los líderes de gimnasio de Kanto, pero no nos interesa el número:

```
$ ./fameChecker -t gym_leader -r kanto --nocount -l
Blaine.html
Daisy_(anime).html
Erika.html
Flint_(Kanto).html
Forrest.html
Giovanni_(anime).html
Koga.html
Lily_(Kanto).html
Lt._Surge.html
Sabrina.html
Violet_(anime).html
```

Si necesitamos tener una idea de cuánta información puede haber sobre un personaje que aparece solo una vez:

```
$ ./fameChecker -t one_time --list --size
Cantidad de archivos encontrados: 258
Bugsy's_Scyther.html (115 KB)
Alex_Davis.html (49 KB)
Alice_Telesu.html (51 KB)
Andrea.html (50 KB)
Andreas.html (58 KB)
Annie_(EP173).html (49 KB)
Arielle.html (47 KB)
Ariene.html (42 KB)
Bailey.html (51 KB)
Band_of_Diglett_thieves.html (49 KB)
Benji's_father.html (56 KB)
Benji.html (51 KB)
Benny_(EP146).html (47 KB)
. . .
```

Si deseamos averiguar cuánta información total hay sobre los personajes principales:

```
$ ./fameChecker -t main --size
Cantidad de archivos encontrados: 10
```

Estructuras del programa

El programa consta de dos **estructuras** principales:

El primero es **character**, que contiene información sobre un personaje. Los campos **name**, **region** y **species** son punteros a caracteres que almacenan el nombre, la región donde el

personaje reside y la especie del personaje, que en este caso puede ser tanto pokemon como trainer (entrenador). El campo **type** es un entero que representa el tipo de aparición (main, recurring, gym_leader, y one_time) y el campo **file_size** es un entero sin signo largo que representa el tamaño del archivo.

Y **search_criteria,** que corresponde al criterio de búsqueda del usuario, en este caso, contiene información sobre los criterios de búsqueda recibidos por el usuario a través de la línea de comandos. Los campos **region**, **species** y **name** son punteros a caracteres que almacenan la región, la especie y el nombre del personaje. El campo **type** es un entero que indica el tipo de aparición del personaje. Los campos **show_count**, **list_files** y **show_size** son enteros que actúan como booleanos para indicar si se debe mostrar la cuenta de resultados obtenidos, la lista de archivos y el tamaño de los archivos, respectivamente.

Por último, tenemos el programa principal **main.c**, que contiene las funciones auxiliares, de gestión de memoria y manipulación de archivos.

Primero se inicializa el programa con los parámetros, los cuales serán procesados por la función **“pase_arguments”** la cual almacenará los datos en la estructura que se encuentra en **“search_criteria.h**” a partir de aquí pasamos a la parte principal que es procesar esa información para obtener los resultados correspondientes por medio de la función **“process_dirctory”** la cual va buscando la informacion especifica de los archivos que se encuentran en la carpeta principal, buscando según la clasificación de la región, el directorio de pokemons y trainer. Esa búsqueda se realiza de manera recursiva para poder obtener el desglose de la información del conjunto de texto del archivo formato **“html”**. De ser necesario también se puede obtener el tamaño de los archivos.

Conclusiones

A partir de la realización de este proyecto, pudimos concluir que el programa nos permitió pulir y poner en práctica los conocimientos adquiridos a lo largo de estas últimas etapas del curso de una manera divertida e incluso más familiar, hicimos uso de varias técnicas y conceptos importantes en la programación en C, incluyendo el uso de estructuras para almacenar datos relacionados, el uso de punteros y la gestión dinámica de memoria, el procesamiento de argumentos de línea de comandos y la automatización de la compilación y construcción del proyecto utilizando un makefile.

Nos aseguramos de que el programa también demostrase buenas prácticas en la organización y estructuración del código, incluyendo la modularización del código en funciones con propósitos específicos y la documentación clara y concisa de las funciones y estructuras.

Por otro lado, creemos que tipos de proyectos como este pueden ser altamente escalables, permitiéndonos utilizarlos para entender más a fondo sobre cómo funcionan los sistemas operativos. Asimismo, un reto que además pudiésemos afrontar es el hecho de poder mejorar la compatibilidad con otros OS que partan del mismo principio o arquitectura similar.

Como palabras finales, fue entretenida la realización de este proyecto, dado que el origen del mismo se basa en un tema bastante familiar, y el exponer conocimientos materializando programas que normalmente veríamos en juegos y en ficción, hace que pongamos mucho más énfasis en el desarrollo de los mismos.