Práctica 2 - Encapsular el acceso a un mainframe

Lunes, 23/11/2020

Eduardo Gimeno - 721615

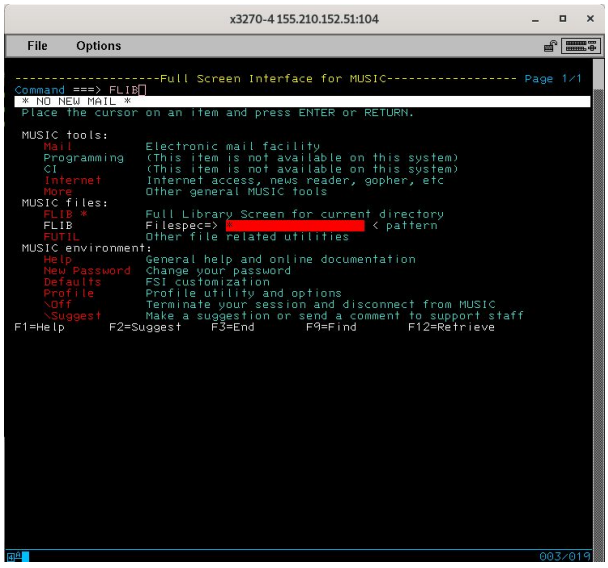
Jorge Turbica - 723883

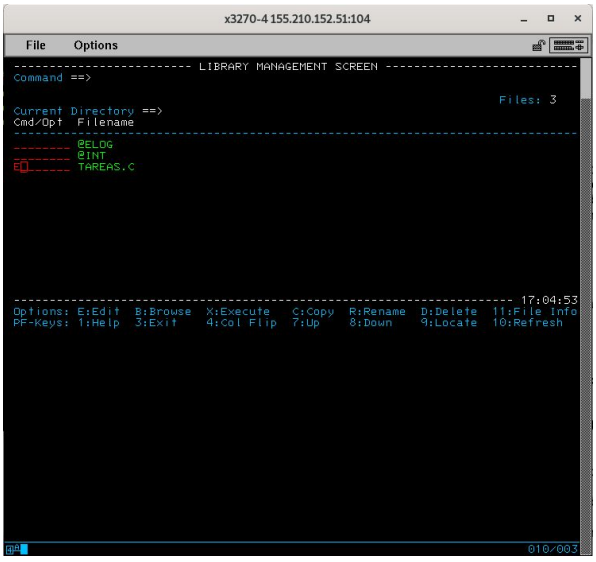
José Navarro - 721230

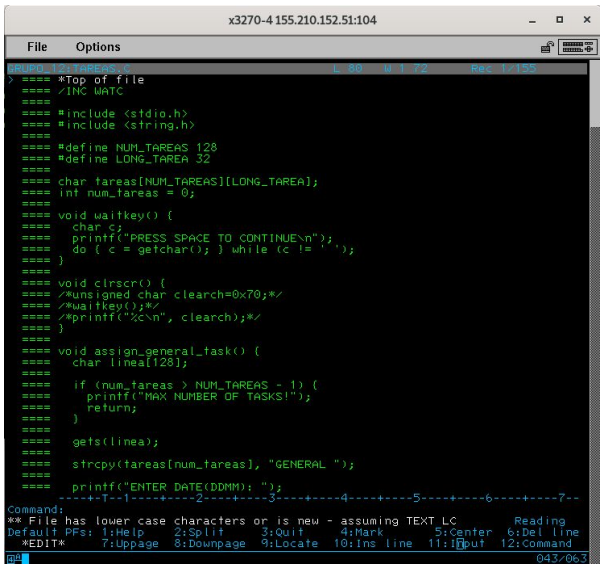
Configuración

En primer lugar, se ha descargado e instalado el emulador x3270. Una vez instalado, se ha probado su funcionamiento siguiendo los pasos indicados en el guión de la práctica. Se ha hecho uso de la aplicación de tareas para ver qué tipo de funcionalidades ofrece. Además se ha visualizado el contenido del fichero para ver los parámetros requeridos en las distintas funciones y poder realizar un análisis más detallado.

Se han seguido los siguientes pasos para poder ver el contenido del fichero tareas.c:







Una vez realizado lo anterior, se procedió a buscar interfaces para el emulador x3270. Se seleccionaron dos como posibles candidatas, Java Robot (vista en clase) y py3270 de Python. Finalmente se seleccionó la interfaz de Python, ya que resultaba más sencilla de utilizar, aunque este lenguaje si no está bien configurado puede presentar diversos problemas. Fue necesario actualizar la versión del lenguaje de la 2.7 a la 3, con los consecuentes problemas que surgieron, principalmente indicar al sistema operativo qué versión debía utilizar. Además también fue necesario actualizar el gestor de paquetes pip, presentando problemas similares.

Una vez seleccionada la interfaz a utilizar, se procedió a buscar una librería, de Python también que permitiera crear GUI, obteniendo como resultado que la más utilizada es Tkinter. Se decidió utilizar esta porque al ser la más utilizada, resultaba sencillo encontrar ejemplos y tutoriales.

Finalmente, el proceso de instalación seguido sería el siguiente:

- Instalar el emulador x3270 dependiendo del sistema operativo.
- Instalar Python-Tkinter en función del sistema operativo (sudo apt-get install python3-tk en Debian).
- Instalar py3270 mediante pip3 install -r requeriments.txt, siendo esta la mejor forma para instalar dependencias de forma automática según la información buscada, el fichero requeriments.txt contiene los nombres de la librerías a instalar.

API para la conexión con el mainframe

Haciendo uso de la librería py3270, se ha creado una API para trabajar con el mainframe a través del emulador (api.py).

Se han definido una serie de funciones para emular las acciones del usuario en la terminal y tratar las respuestas del mainframe que se muestran en la misma. Se han creado funciones para la conexión y desconexión con el mainframe, login, ejecución del programa tareas.c, limpieza de pantalla, introducir parámetros en la terminal, volver al menú principal, añadir tarea, listar tareas y espera para que la pantalla esté lista.

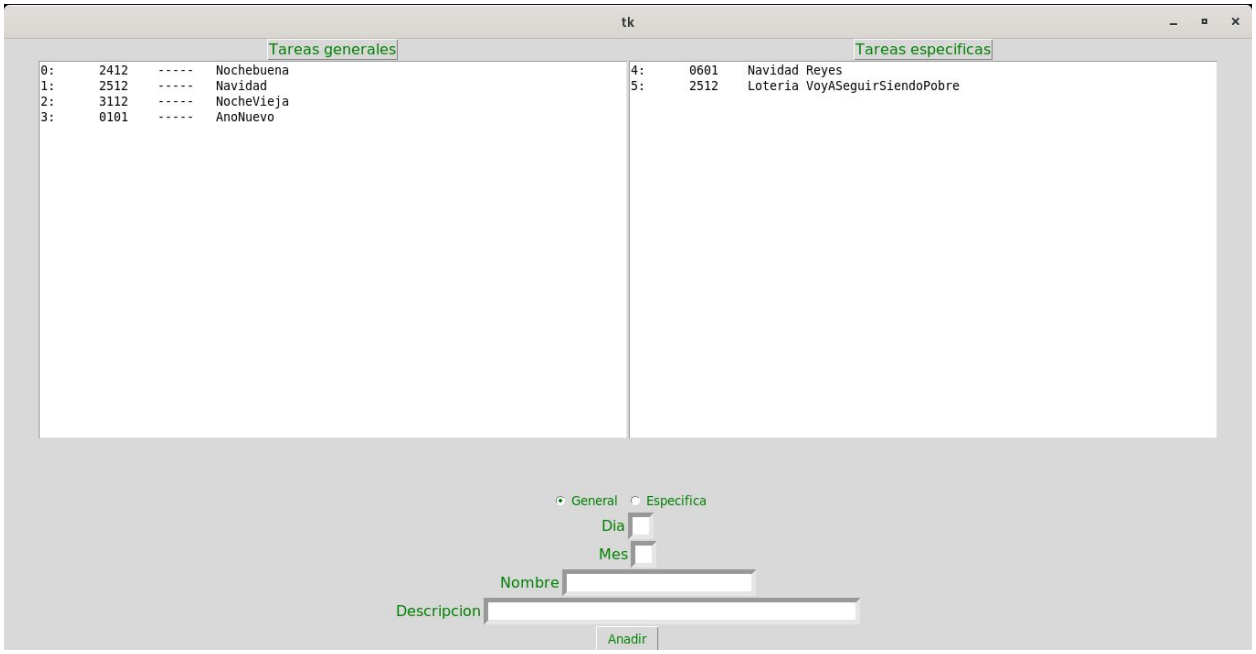
La función ofrecida por la librería py3270 para esperar a que la pantalla esté lista presentaba problemas y muchas veces no se obtenían los resultados deseados, se mostraba un mensaje de error indicando que el teclado estaba bloqueado. Por tanto, en la función creada para la espera de la pantalla, ha sido necesario añadir una espera de un determinado número de segundos y así garantizar que el teclado no quedaba bloqueado. Se ha observado que este número de segundos varía en función de cómo se encuentre la conexión con el mainframe, si este tiene mucho tráfico y tarda en responder, es necesario aumentar dichos segundos de espera.

Para poder leer determinadas respuestas siempre en las mismas posiciones de la terminal, como es el caso de listar las tareas, se ha forzado la limpieza de la pantalla escribiendo órdenes de forma consecutiva, que no afectan al estado de la aplicación (pulsar enter), hasta que se vuelve a escribir en la parte superior de la terminal.

Como hay comportamientos comunes en la mayoría de las operaciones, como es esperar a que la pantalla esté lista o volver al menú principal y limpiar la pantalla en el caso de añadir una tarea y listar las tareas, se ha hecho uso de los decorators de Python para mejorar la legibilidad del código.

Nueva GUI

Se ha hecho uso de la librería Tkinter para crear una nueva GUI. Se ha dividido la pantalla en distintos frames. Partiendo de un único frame (root), se ha dividido en dos, uno para los listados de las tareas y otro para la sección de los datos de una nueva tarea. El frame para los listados se ha dividido a su vez en dos, uno para el listado de tareas generales y otro para las específicas. El frame para los datos de una nueva tarea se ha dividido en cinco, uno para el tipo de tarea, otro para el día, otro para el mes, otro para el nombre y otro para la descripción. Se han añadido las etiquetas y campos de texto necesarios en cada frame mencionado, ya sea para mostrar o introducir datos. Al pulsar el botón Anadir, se invoca a la función addTask, la cual hace uso de las operaciones para añadir y listar tareas ofrecidas por la API que se ha creado, añadiendo en primer lugar la nueva tarea y actualizando los listados después.



Distribución del trabajo

Tarea	Eduardo	Jorge	José
Instalación y configuración			
API			
GUI			
Informe			

Bibliografía

py3270 0.3.5 [https://pypi.org/project/py3270/]

Python GUI - tkinter [https://www.geeksforgeeks.org/python-gui-tkinter/]

Tkinter - Installation or Setup [https://riptutorial.com/tkinter/example/3206/installation-or-setup]

Cómo instalar paquetes Python con requeriments.txt [https://rukbottoland.com/blog/como-instalar-paquetes-python-con-requirementstxt/]

Why you need decorators in your Python code
[https://medium.com/better-programming/why-you-need-decorators-in-your-python-code-df12d43eac9c]