CSSE1001

16/10/2013

Assignment 3 – Design Document

Name: Thuan Song TEOH
Student Number: 43068052
Project Title: Android Explorer

1. Description

The main objective of this project is to create a **file manager** which runs on **Windows and Linux** that **manages files in Android devices**, utilising the **Android Debugging Bridge (ADB)**. The main reason for this is because users of Android devices often **face USB connectivity issues** due to missing drivers or USB connection protocol incompatibility. Linux users will understand this better as MTP (which was meant for Windows, but unfortunately the main protocol used by recent and upcoming Android devices) might be hard to setup if inexperienced. Therefore, by using ADB, this problem can be easily solved as the drivers are already made available by Google; therefore it can be easily obtained compared to drivers by OEMs. Dropbox sync and ADB core functions were coded in a way that they could be reused as a **command line tool** as well.

2. User Interface (UI)

The UI is like most programs, it is window based, i.e. a new window/dialog will be called up for each main task given to it. Figure 1 below shows the main window of the program:

Figure 1: Main window

As can be seen from Figure 1, there are 4 main icons, and the title below the icons already shows the main functions of the program.

1. **File Manager**

   This is the core of the program; user will be given a choice of using the file manager as single panel or dual panels. Operations in the single panel version will be executed mainly by **mouse clicks and keyboard input**. When launched, it checks for a connected device. The UI for the single panel is as below:

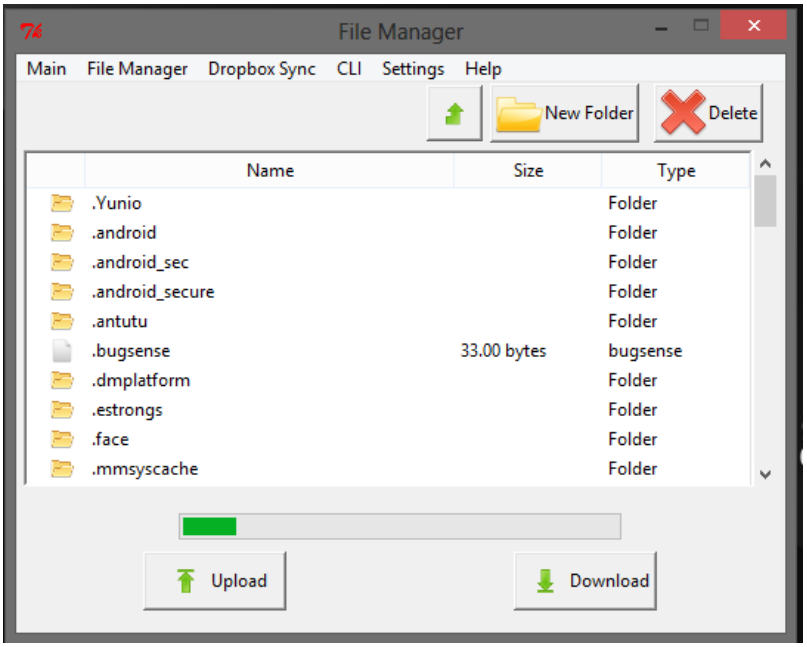

Figure 2: Single panel

- Tree view widget the middle: Display files in the device.
- Column headers: When clicked, will sort files accordingly (ascending and descending)
- Green bar: Progress bar.
- Buttons: Functions explained by the label and icons.

For dual panel, the actions were proposed to be similar to the single panel. The layout was proposed to be as below:

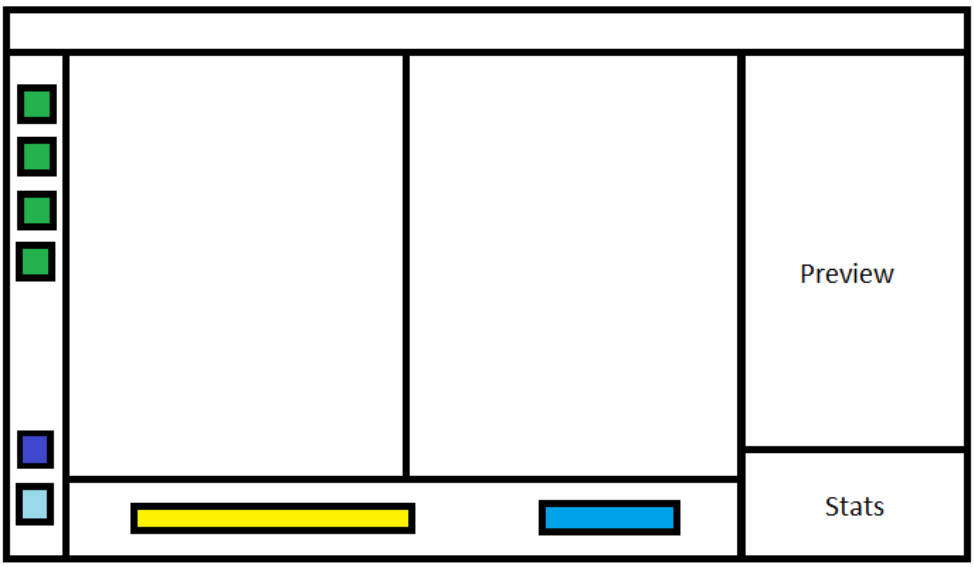

Figure 3: Dual panel

- Blank canvas at the left: Display files in the computer.
- Blank canvas at the right: Display files in the device.
- Light blue and dark blue buttons at the left: Toggle between thumbnail view and list view.
- Green buttons: When clicked, will search for files corresponding to categories (determined by the button), group them and display them instead of every single file in the device. The categories will be: images, videos, music and documents.
- Rightmost canvas: Preview for media files, a mini music player will appear if a sound file (.mp3, .wav) is selected, and images will be displayed if an image file (.jpg, .png, .gif) is selected.
- Canvas at bottom right: Show space remaining in device.
- Yellow bar: Progress bar.
- Blue button on bottom right: Upload files, another dialog will pop up for user to select file from computer to upload to device.

2. **Dropbox Sync**

When launched, will prompt for an authorisation code in a dialog (can be set to auto logout). The window is a simple one as shown below:

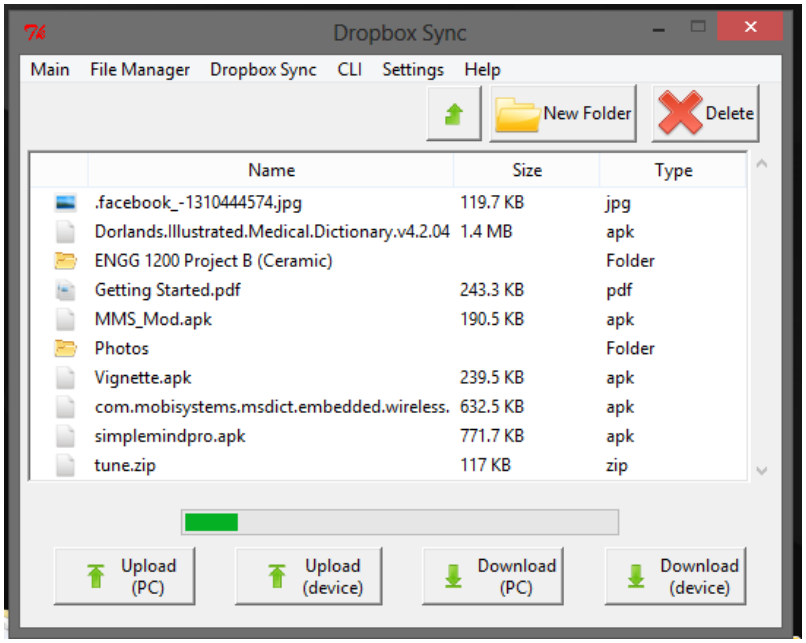

Figure 4: Dropbox sync window

- Tree view in the middle: Display files in Dropbox account.
- Buttons: Functions explained by labels and icons.
- Green bar: Progress bar

3. **Settings**

This will be the place where user can customise the behaviour of the program.

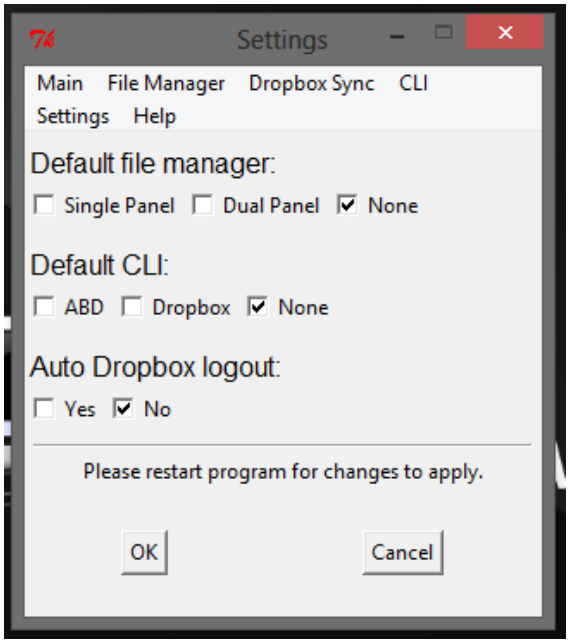

Figure 5: Settings window

The checkboxes determine the default behaviour of the program. If no default file manager or CLI is chosen, the users will be prompted to select one via a dialog box when running from the main menu.

4. **General**

   As seen from Figure 1 to 5, there is a menubar on the top of the window, which is the menu bar. The entries are as below:

   - Main
     - Exit
   - File Manager
     - Single Panel
     - Dual Panel
   - Dropbox Sync
   - CLI
     - ABD
     - Dropbox
   - Settings
   - Help
     - Help
     - About

   For the File Manager and Dropbox Sync windows, there is a context menu when the user performs a right mouse click. The context menu is as below:

   - Refresh
   - New folder
   - Rename
   - Delete

   Also in the file managers (ADB and Dropbox), there are special keyboard commands. The Ctrl key will used for multi select, i.e. users can select multiple files when the Ctrl key is held down. The F5 key can be used to refresh a window. The Del key will delete files. The F2 key can be used for renaming files.

3. Design

The program consists of:

1. ui/main.py
2. ui/func.py
3. core/core_host.py
4. core/core_target.py
5. core/core_cli.py
6. core/core_drpbx.py
7. file_man.py
8. drpbx_ui.py

9. settings.py
10. run.pyw
11. cli_target.py
12. cli_drpbx.py

Description template:

Class (descriptions and constructors: name)			
Method Name	Method Parameters (*type* name)	Method Description	Return Variables (*type* name)

MainUI *in ui/main.py* The main GUI class where the GUI items such as menu bars and context menu will be hosted. Plugins can use this class directly or inherit from this class to ensure uniformity in UI design. (Tkinter)			
__init__	*instance* master, *instance* func	Initialises MainUI.	
create_menubar		Creates the menu bar.	
create_context_menu		Creates the context menu.	
_pop_context	*object* event	Callback function to reposition the context menu.	

UIFunc *in ui/func.py* Main functions of components in MainUI. (Tkinter)			
__init__	*instance* master, *instance* remote, *instance* tree	Initialises UIFunc.	
_execute	*string/list* cmd	Carries out a command without showing a shell.	
new_fol		Creates a new folder (called from context menu).	
rename		Renames files/folder (called from context menu).	
delete		Deletes files/folder (called from context menu).	
refresh		Refreshes tree view (called from context menu).	
close		Closes window (called from menu	

		bar).	
single		Opens a single panel file manager.	
dual		Opens a dual panel file manager.	
drpbx		Opens Dropbox sync.	
cli_target		Launches ADB CLI in another shell.	
cli_drpbx		Launches Dropbox CLI in another shell.	
settings		Opens Settings window (called from menu bar).	
helpfile		Opens help file (called from menu bar).	
about		Display program information (called from menu bar).	
create_new_dialog	*string* title, *string* text, *int* size, *function* cmd	Creates a small dialog box with an entry widget.	(*object* entry, *object* win)

Core_host *in core/core_host.py* The class where all the commands to carry out actions on the computer is hosted.			
__init__		Initialises Core_host.	
ls	*string* folder	Lists files in current directory or specific directory (if given). Also checks file size and if file is directory.	*dict* out
pwd		Returns current path.	*string* self._cur_path
cd	*string* path	Changes current directory to pth.	
cp	*string* _from, *string* _to	Copies file to target.	
mv	*string* _from, *string* _to	Moves files to target.	
mkdir	*string* path	Creates a new folder.	
rm	*string* path	Deletes files.	
_convert	*int* num	Converts given number into bytes, KB, MB, GB or TB.	*string* num

Core_target *in core/core_target.py* The class where all the commands to carry out actions on the device is hosted.			
__init__		Initialises	

		Core_target.	
_execute	*string/list* cmd	Executes shell commands and returns the output as a list.	*list* lines
check_connect		Check whether the device is connected	
connect		Connects to device.	
ls	*string* folder	Returns a list of files in the current directory.	*list* file_list
pwd		Returns current path.	*string* cur_pth
cd	*string* path	Changes current directory to path.	
cp	*string* _from, *string* _to	Copies file to target.	
mv	*string* _from, *string* _to	Moves files to target.	
rm	*string* path	Deletes files.	
mkdir	*string* path	Creates a new folder.	
upload	*string* _from, *string* _to	Uploads files to target (in device).	
download	*string* _from, *string* _to	Downloads files to target (in computer).	
env		Get path of device storage.	
df		Displays disk usage..	*dict* out
_convert	*int* num	Converts given number into bytes, KB, MB, GB or TB.	*string* num
_check_error	*list* out	Raises an exception if there is an error when executing shell commands.	
_check_is_dir	*string* path	Checks if the given path is a folder or not (targeted at symlinks).	*dict*

ADBError *in core/core_target.py* Exception for ADB action errors. (Exception)

Core_drpbx *in core/core_drpbx.py* The class where all the commands to carry out actions on Dropbox is hosted.			
__init__		Initialises core_drpbx.	
get_auth_url		Gets authorisation	string

		URL.	self._flow.start()
login	*string* code	Login using authorisation code.	
logout		Logouts from Dropbox.	
upload	*string* _from, *string* _to	Uploads files to Dropbox.	
download	*string* _from, *string* _to	Downloads files to target (in computer).	
pwd	*string* path	Returns current path.	*string* self._cur_path
ls	*string* folder	Returns a list of files in the current directory.	*dict* out
cd	*string* path	Changes current directory to path.	
mv	*string* _from, *string* _to	Moves files to target.	
mkdir	*string* path	Creates a new folder.	
rm	*string* path	Deletes files.	
acc_info		Displays quota info	*list* out
_convert	*int* num	Converts given number into bytes, KB, MB, GB or TB.	*string* num

Core_cli *in core/core_cli.py* Main command line inteface (CLI) class. Other CLIs will inherit this. (Cmd.cmd)			
do_ls	*string* folder	Lists files in current directory or specific directory (if given). Also checks file size and if file is directory.	
do_lls	*string* args	Lists file in local	
do_mkdir	*string* args	Creates folder in remote	
do_rm	*string* args	Deletes a file/folder in remote.	
do_cp	*string* _from, *string* _to	Copies a remote file/folder to another remote location.	
do_mv	*string* _from, *string* _to	Moves a remote file/folder to another remote location.	
do_pwd		Shows current device working directory.	
do_lpwd		Shows current local working directory.	
do_cd	*string* args	Changes current	

		remote working directory	
do_lcd	*string* args	Changes current local working directory	
do_put	*string* _from, *string* _to	Uploads a file from local to remote	
do_get	*string* _from, *string* _to	Downloads a file from remote to local	
do_help	*string* args	List available commands with "help" or detailed help with "help cmd".	
emptyline		Default behaviour: Runs previous command if hit Enter with no commands. Since we are dealing with files, disable to prevent unintentional actions.	
parseline		Default parseline is not flexible enough, does not take account of quotes, so use shlex instead	*tuple*

Cli_drpbx *in cli_drpbx.py* Command line inteface for Dropbox sync. (Core_cli)			
__init__		Initialises command line interface.	
do_login		Login into a Dropbox account.	
do_logout		Logout from a Dropbox account.	
do_status		Get quota info of Dropbox account.	
do_exit		Terminates program.	
do_EOF		Hit Ctrl+D to exit program.	

Cli_target *in cli_target.py* Command line inteface for ADB sync. (Core_cli)			
__init__		Initialises command line interface.	
do_connect		Connect to a device.	

do_status		Get disk usage info of device.	
do_exit		Terminates program.	
do_EOF		Hit Ctrl+D to exit program.	

cli_deco (*bool* connected) *in cli_drpbx.py and cli_target.py*
Decorator for command line. Handles exception and prints in a fixed format.

FileMan *in file_man.py*
The class for the file manager GUI.
(Tkinter)

__init__		Initialises FileMan.	
_create_top_widgets		Creates the buttons on top.	
_create_bottom_widgets		Creates the buttons and progrss bar.	
_create_tree		Creates a tree view of files.	
_build_tree		Adds items to the tree view.	
_prompt_connect		Prompts user to connect to device.	
_cd	*event* event	Change current directory.	
_cd_up		Goes up one directory.	
_rm		Deletes files/folders.	
_mkdir		Creates a new folder.	
_upload		Uploads file from PC.	
_download		Downloads file to PC.	
_refresh	*event* event	Refreshes tree view.	
_remove	*event* event	Deletes files/folders.	
_rename	*event* event	Renames files/folders.	
_sortby	*instance* tree, *string* col, *int* descending		

DrpbxUI *in drpbx_ui.py*
The class for the Dropbox GUI.
(Tkinter)

__init__		Initialises DrpbxUI.	
_create_top_widgets		Creates the buttons on top.	

_create_bottom_widgets		Creates the buttons and progrss bar.	
_create_tree		Creates a tree view of files.	
_build_tree		Adds items to the tree view.	
_prompt_connect		Instructions for logging into Dropbox.	
_cd	*event* event	Change current directory.	
_cd_up		Goes up one directory.	
_rm		Deletes files/folders.	
_mkdir		Creates a new folder.	
_upload_local		Uploads file from PC.	
_upload_device		Uploads file from device.	
_download_local		Downloads file to PC.	
_download_device		Downloads file to device.	
_refresh	*event* event	Refreshes tree view.	
_remove	*event* event	Deletes files/folders.	
_rename	*event* event	Renames files/folders.	
_close		If specified, logout of Dropbox before exiting.	
_sortby	*instance* tree, *string* col, *int* descending		

Func *in drpbx_ui.py* The class for the settings GUI. (UIFunc)			
__init__	*instance* master, *string* cfg, *instance* remote, *instance* tree	Initialises Func.	
close		Logouts from account if specified before exiting.	

Settings *in settings.py* The class for the settings GUI.

(Tkinter)			
__init__		Initialises Settings..	
_create_widgets		Saves changes to a config file.	
_ok		Saves changes to a config file.	
_cancel		Cancels operation and closes window.	

StartPage *in run.pyw* The main class to execute the program. (Tkinter)			
__init__		Initialises StartPage.	
_create_widgets		Creates the buttons and banner.	
_file_man		Callback function for the File Manager button.	
_cli		Callback function for the CLI button.	
_new_dialog	*string* txt1, *string* txt2, *function* cmd	Creates a new dialog with radio buttons.	*object* win

4. Support Modules

- Tkinter (GUI)
- Python Imaging Library (PIL)
- SendKeys (for simulating key press in Windows): https://pypi.python.org/pypi/SendKeys
- xdotool (for simulating key press in Linux, not a Python library): http://manpages.ubuntu.com/manpages/lucid/man1/xdotool.1.html
- Dropbox Python SDK (for Dropbox sync): https://www.dropbox.com/developers/core/sdks/python
- Android Debugging Bridge (to access files and Android shell): http://developer.android.com/tools/help/adb.html