



Assignment 1

[Objectives](#) | [Assignment 1](#) | [Any Questions?](#) | [Submitting your Program](#)

Due Date: 22 Aug

1. Objectives

The objectives of this assignment are to:

1. Build upon Python programming skills begun in Lab Session 1 ([Objective 6](#));
2. Develop skills in concurrent programming ([Objective 3](#)); and
3. Write a program that will be a useful demonstration of your learning in this unit ([Objective 8](#)).

The exercise is designed to be both fun, and useful! - you should be able to use it as a convenient way of using your background window to run through your favourite digital photographs.

2. Assignment 1

You are to write a Python program to display a (background) image, and at the same time interact with the user. For this, you will need to use Python threads. You will need to visit the library page on [Threads](#) (or [Threading](#)) to see how to use them. Note that you may use either the low-level threads version in section 7.4, or the high-level version threading version found at section 7.5.

2.1 Display Image

To display a background image on Mac OS X, one can use the following shell script:

"SetBackground-MacOSX.sh" 2.1 =

```
#!/bin/sh
# collect parameter to script, the image pathname
IMAGE=$1

# run apple script to change background picture
/usr/bin/osascript <<END
tell application "Finder"
    set myFile to POSIX file "$IMAGE" as string
    set desktop picture to file myFile
end tell
END
```

Note that this works on Mac OS X only - bonus marks to the first five students who send me (ajh AT csse DOT monash DOT edu DOT au) an equivalent script to do the same thing under Windows/Linux/your favourite operating system. (5 marks to the first student, 4 to the second, 3 to the third, etc..) Note that it **must** display as a background image, that is, it must not appear in a separate window.

Jeff Parsons is the first to respond with a working solution for Linux/Gnome - congratulations, Jeff! 5 bonus marks for you!

"SetBackground-Gnome.sh" 2.2 =

```
#!/bin/bash
# collect parameter to script, the image pathname
IMAGE=$1

# define the desktop wallpaper filename
BG=/desktop/gnome/background/picture_filename

# perform the setting
gconftool-2 -t str -s $BG $IMAGE
```

where

- t str specifies the type of the setting is a string
- s specifies that we are SETting the value

Just after I posted that message, another four appeared in my mailbox! Here they are (note that the bonus marks are now all gone 😊):

"SetBackground-Windows.py" 2.3 =

```
def setWallpaperFromBMP(imagepath):
    """changes the wallpaper using 'imagepath'"""
    SPIF_UPDATEINIFILE = hex(1)
    SPI_SETDESKWALLPAPER = 20
    SPIF_SENDWININICHANGE = hex(2)
    ctypes.windll.user32.SystemParametersInfoA(SPI_SETDESKWALLPAPER, 0, imagepath , SPIF_UPDATEINIFILE or SPIF_SENDWININICHANGE)

if __name__ == "__main__":
    #file path is read from first argument after module name
    setWallpaperFromBMP(sys.argv[1])
```

Note that this can either be run as a script:

<run as script 2.4> =

```
python SetBackground-Windows.py c:\windows\greenstone.bmp
```

or it can be imported as a python module, and called directly:

<run as subroutine 2.5> =

```
import SetBackground-Windows
...
setWallpaperFromBMP(imagepathname)
```

know.

Sohrab adds:

Warning: You do not need to understand the following section: it is included for information and interest only.

I had time to test the code for windows implementation of the wallpaper changer and I can confirm it does not work with any image format but bmp.

However, I've written a second version of the program (see below) that handles all of the common image types, in case you are interested. it does so by creating a temporary bmp file in the modules folder and piping that into win32 API. this program uses python imaging library (PIL) which, i believe, is not supplied with the standard python packages but can be freely downloaded from <http://www.pythonware.com/products/pil/>

I've also modified the program so the the wallpaper change is not permanent; i.e. after a reboot, the desktop wallpaper is reverted to the original (the image that was set before running this program). I thought it made more sense in the context of our exercise.

"SetBackground-Windows2.py" 2.6 =

```
import ctypes, sys, Image, os

def setWallpaper(imagepath):
    """changes the wallpaper using 'imagepath'"""

    SPI_SETDESKWALLPAPER = 20                #win32 API contant
    default_path = sys.path[0]                #path to save temporary bmp file:
                                              #currently the path of the module

    #adding \ at the end, in case it's missing
    if not default_path[-1] == "\\": default_path = default_path + "\\"

    try:
        Image.open(imagepath).save(default_path + "temp.bmp")
        imagepath = default_path + "temp.bmp"
    except IOError:
        raise "unsupported image format"

    #win32 API call
    ctypes.windll.user32.SystemParametersInfoA(SPI_SETDESKWALLPAPER, 0, imagepath , 0)

    #delete temp image
    #os.remove(imagepath)

if __name__ == "__main__":
    #file path is read from first argument after module name
    setWallpaper(sys.argv[1])
```

Now read on: →

Also to Rian McGuire, who sent in a similar solution to Jeff Parson's one above. 3 bonus marks to Rian!

Dmitri Nikulin has sent in a neat generic solution, which you can find on the [fsetbg web page](#). 2 bonus marks to Dmitri!

Finally, Victoria Colquhoun collects the last bonus mark, for her solution which was also for Gnome.

My thanks to the 5 students who rose to the challenge!

2.2 The Database

Images to be displayed are determined by reading a database. The database file consists of a set of text lines, where each line contains the full file name (*pathname*) of the image, and a time value *displaytime* (in seconds). The two fields are separated by one or more blanks. The first line of the file contains just a single integer number, which is the ordinal number of the next image to be displayed.

The database must be updated as images are displayed, and a separate thread is to be used to perform the update, so that it runs concurrently with other activities in the program. The database should be updated whenever any information it contains is altered (including the next image number).

Here is a sample of what the database might look like:

"database.txt" 2.7 =

```
2
/home/ajh/photos/trains/puffingbilly/NA6-1.jpg 60
/home/ajh/photos/trains/puffingbilly/NA7-2.jpg 120
/home/ajh/photos/mallacoota/boattrip-1.JPG 90
/home/ajh/photos/adelaide/family-3.JPG 30
```

Warning: This exercise should be completed only after all other basic sections have been completed, and are working satisfactorily.

Add a feature to your program to check before writing the database that it hasn't been (mutually) updated elsewhere. If the file has been modified, issue a warning to the user, and ask what should be done. Possibilities are: overwrite the file; abandon the update; abandon the update and read the new external database; etc..

Flag your addition in the code with a line `# CHECK DB UNMODIFIED`.

Now read on: →

2.3 Display Management

The background display is set in turn to each image *pathname* in the database and maintained for a fixed time, the time determined by the *displaytime* value stored in the database. At the conclusion of the time, or when a user command requires, a new image is chosen and displayed for the appropriate amount of time. The index number of the image is updated, and the database is marked as changed.

2.4 The User Interface

This section of the program is responsible for interacting with the user, and also runs asynchronously with the other sections.

The user input consists of a series of commands entered on standard input (`sys.stdin`). The commands are normally single characters, but make take parameters. Multiple commands may be entered on a single input line, but they must be separated by whitespace. You must include at least the following commands:

- n display the next image immediately (cancelling the time on the currently displayed image).
- p display the previous image immediately (cancelling the time on the currently displayed image).
- t d+ set the time for the current image to be displayed to d+ seconds (d+ means one or more digits 0-9).
- g d+ go to image number d+ in the database and display.
- q quit the program

Warning: This exercise should be completed only after all other basic sections have been completed, and are working satisfactorily.

You may add other commands if you wish.

Now read on: →

2.5 Overall Program Management

This section is the main body of the program, and is responsible for setting up the various subcomponents, initializing the threads, and gathering everything together upon the user entering the quit command.

3. Any Questions?

If there is anything unclear, or something you think is needed is omitted, ask your question in the [Anonymous Feedback Page](#) and it will be answered as soon as possible.

4. Submitting your Program

Your program is to be submitted through MUSO as a tar file (uncompressed). The tar file should be named `12345678.tar`, where 12345678 is your student ID number. It should contain `display.py`, the complete Python text of your program, in runnable form, together with a pdf file `discussion.pdf`. **Do not submit these files directly as they will be discarded!**

and please, no Word files!!

The `discussion.pdf` should contain a short (no more than an A4 page) summary with the following headings: Aims of this Exercise; Problems Encountered; Solutions Adopted; and Testing Strategies. Note that the Aims of the Exercise is **not** the same thing as the Objectives of the Exercise!

Warning: This exercise should be completed only after all other basic sections have been completed, and are working satisfactorily.

You may also wish to add a section on Future Enhancements (not counted in the one page limit).

Now read on: →

Your display program code must be laid out according to the following schema:

"display.py" 4.1 =

```
# d i s p l a y . p y
#
# Your name, Your student ID number
#
# other archival information that may you wish to include, for
# example, date, version number, etc..
...
# SECTION 0 : Global Data
#         library imports, data shared across the program, etc.
...
# SECTION 1 : Display Image
#         code relevant to displaying the actual image
...
# SECTION 2 : The Database
#         all functions/classes pertinent to managing the database
...
# SECTION 3 : Display Management
#         all functions/classes pertinent to managing
#         the display timing and scheduling
...
# SECTION 4 : The User Interface
#         all functions/classes pertinent to running the user interface
...
# SECTION 5 : Program Management
#         the main body of the program
...

```

Document History

20070801:140456 1.0.4 ajh added other set background solutions
 20070801:110124 1.0.3 ajh added Jeff Parson's Linux/Gnome Setbackground script
 20070731:144944 1.0.2 ajh released to csse server
 20070731:074014 1.0.1 ajh first draft
 20070718:160810 1.0.0 ajh stub only

This page maintained by John Hurst.

Copyright [Monash University Copyright Policy](#)

Generated at 20070808:1354 from an XML file modified on 20070807:1538

Maintainer use only; not generally accessible: [Local Server](#) [Work Server](#) [CSSE Server](#)

58 accesses since
18 Jul 2007

