Indian Institute of Technology Palakkad

Department of Computer Science and Engineering

Summer Internship Report

Place of Internship: Hamon Technologies LLP, 1st floor, Behind MIMS hospital, Near popular automobiles, Easwaramangalam road, Govindapuram P.O. , Kozhikode

Period of Internship: from 10th May to 30th June 2017 for a period of 8 weeks

Internship Topics:

1. UZBL replacement using pygame
2. Porting the excellent Python library psutil to C (cpslib)

Name and Designation of the Mentor: Noufal Ibrahim, The founder of Hamon.

NOUFAL IBRAHIM
THE FOUNDER OF HAMON
HAMON TECHNOLOGIES LLP

July 14, 2017
Authored by: Libin N George
Roll No: 111501015

Indian Institute of Technology Palakkad

SUMMER INTERNSHIP REPORT

1. UZBL replacement using pygame

AIM

The chief aim is to remove UZBL from railway ticket info system so that we can run the entire application without X-windows. Our Aim is to create a pygame program or a module which can be used to mimic the UZBL (Stage 1 of the above Main Project). It can read data at regular (small) intervals from a web service (which is the existing part) and then display that on a pygame powered screen.

Requirements

The layout should be customisable according to the screen size. The total screen size is 1080x1920 (vertical). Ticketing info layout will have two configurations 70-30 ratio (70% advertisements and 30% ticket information) and 75-25 ratio. Layout should be configurable for easy changes like colour, founts, coordinates etc. Language information, branch phone number .etc in the layout needs to be in a config file. (70-30 ratio layout: 1080x576, 75-25 ratio layout: 1080x480)

Summary

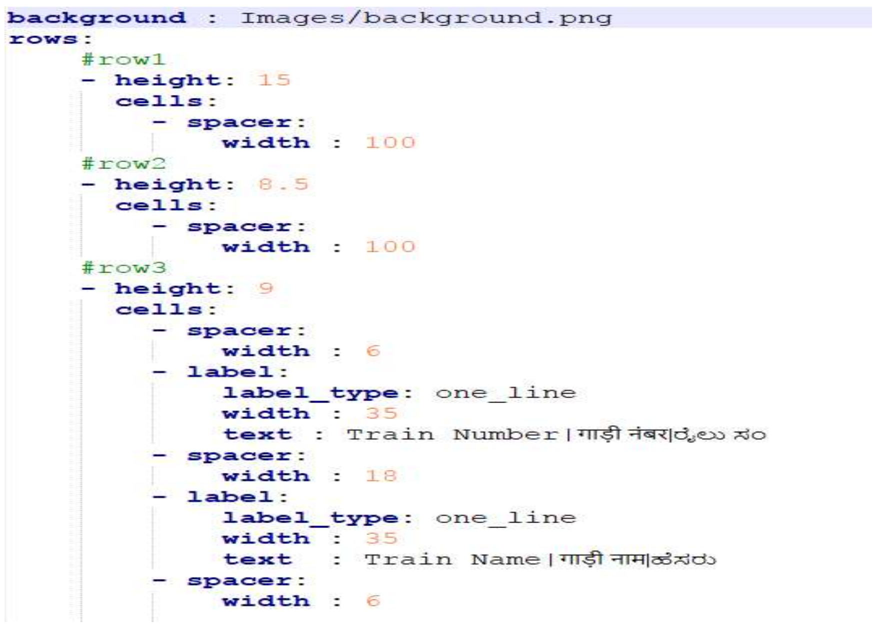
The Client, Vyoma Technologies uses a web server and a java script to display info to the bottom of the Screen using a Raspberry pie. The web server populates an internal dictionary using information from the serial port. A URL is used to fetch the internal dictionary in JSON format which is used by UZBL to display ticket information. The Main aim of the project is to replace UZBL which requires X-Windows with a pygame based interface which could run without X-Windows.

Pygame is a Free and Open Source python programming language library for making multimedia applications like games built on top of the excellent SDL library. Like SDL, pygame is highly portable and runs on nearly every platform and operating system. Pygame uses either opengl, directx, windib, X11, Linux frame buffer and many other different backends.

Config file contains values for screen width, screen height, fonts for different languages and default values for some parameter like font size, font color, scroll speed etc. These default values can be changed for individual widgets in the *layout file*.

Layout files (*.yaml)

Layout file is a dictionary containing keys background and rows. The value of key background is the path to background image. The value associated with the key rows is a list of row arguments. Each row argument is a dictionary containing keys height and cells (the details of the cells in the row).The key cells contain the list of widgets each of which occupies one cell. The widget named spacer, label, image and text field are available. All widths and height in the layout file should be percentage (giving compatibility with different screen size). Optional arguments like font colour, font size etc.



```
background : Images/background.png
rows:
    #row1
    - height: 15
      cells:
        - spacer:
            width : 100
    #row2
    - height: 8.5
      cells:
        - spacer:
            width : 100
    #row3
    - height: 9
      cells:
        - spacer:
            width : 6
        - label:
            label_type: one_line
            width : 35
            text : Train Number | गाड़ी नंबर|ರೈಲು ಸಂ
        - spacer:
            width : 18
        - label:
            label_type: one_line
            width : 35
            text   : Train Name | गाड़ी नाम|ಹೆಸರು
        - spacer:
            width : 6
```

FIGURE 1: LAYOUT FILE (*.YAML)

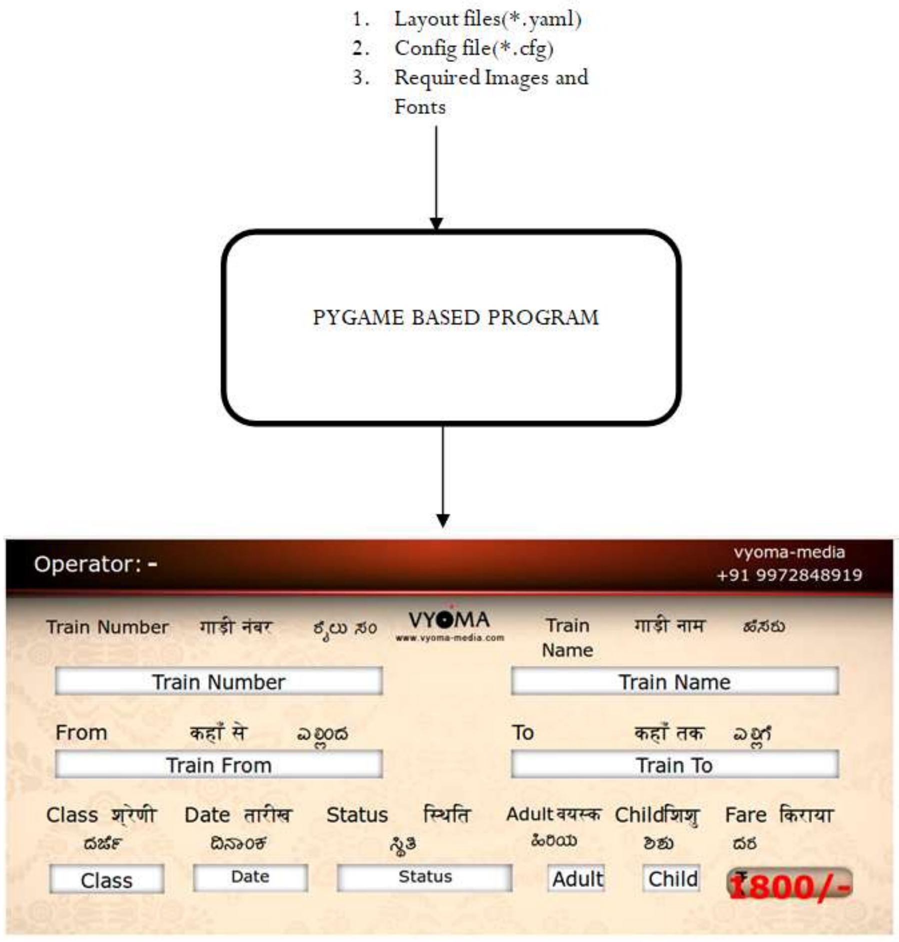


FIGURE 2: FLOWCHART MODEL

Learning outcomes

- Programming skills in python
- Learned to code according to PEP-8 style for python code (used in pylint)

2. Porting the excellent Python library psutil to C

AIM

Port the excellent Python library psutil to C.

1. Port net Connections API from psutil to cpslib in Linux platform
2. Port APIs from psutil to cpslib for Windows Platform

Summary

Cpslib (https://github.com/hamon-in/cpslib) is an attempt to port the excellent Python library psutil to C and then write wrappers for it in other languages (like Rust). There are 3 platform specific files which implement all the API calls. They will hide the details of the actual OS level function which is called to get the job done. The top level header file pslib.h has all the constants and structures necessary to provide the cross platform API.

Port net Connections API (Linux platform)

In Linux, the */proc/net/tcp4, /proc/net/tcp6, /proc/net/udp4, /proc/net/tcp6, /proc/net/unix* files contain information about TCP4, TCP6, UDP4, UDP6, UNIX connections respectively. These files contain a list of connections with its index, local address, remote address, status, inode etc. For UNIX connections there is no remote address. In the API we have to parse the file lines to get the required fields. Using inode number we have to find PID and file descriptor. This could be done by examining the */proc* file (numbers in the folder name shows PID while links inside each folder shows file descriptor with link to associated inode). Another challenge is to convert Address from this file to Human readable form (e.g. "0000000000000000FFFF00000100007F:9E49" -> ("::ffff:127.0.0.1", 40521)). This can be done using inet_ntop function.

Port APIs from psutil to cpslib for Windows Platform

The APIs called disk_io_counters, disk_partitions, disk_usage, get_users, get_cpu_times_percent, get_process etc. have been ported from psutil to cpslib. Disk_io_counters information about read and write rates for each physical disk. It uses DeviceIoControl function to Query the system about IO disk rates. Disk_partitions uses GetLogicalDriveStrings function to get Disk Labels and uses it to find the information about disk partitions. Disk_usage returns total, free, used for disk containing the path. Get_users API gives username, login time, and user address for current user. WTSQuerySessionInformation and WinStationQueryInformation are used to get the Information regarding User. Get_cpu_times_percent provides utilisation (difference) percent for CPU (total or per CPU) form given CPU times structure. Get_process returns the details of the process (PID, parent PID, name, exe, command line, etc.) using given PID. Several other APIs are also implemented (by Athul).

We have used wrapper which is created using CFFI to test APIs. The tests were done using pytest and were successful as expected. Another Problem is undetected Memory Leaks and Memory corruption which could cause crashing under certain conditions.

Memory leaks, defined as the failure to correctly deallocate memory that was previously allocated, are among the most subtle and hard-to-detect bugs in C/C++ applications. A small memory leak might not be noticed at first, but over time, a progressive memory leak can cause symptoms that range from decreased performance to crashing when the application runs out of memory. Worse, a leaking application that uses up all available memory can cause another application to crash, creating confusion as to which application is responsible. Even seemingly harmless memory leaks might be symptomatic of other problems that should be corrected.

The C Run-Time (CRT) libraries provide you with the means for detecting and identifying memory leaks. To enable the debug heap functions, include crtdbg.h after including stdlib.h. Defining _CRTDBG_MAP_ALLOC before this enable detailed debug report. Defining _DEBUG will enable the debug versions of malloc, free, etc. in the crtdbg.h. After enabling the debug heap functions by using these statements, place a call to *_CrtDumpMemoryLeaks* before an application exit point to display a memory-leak report. Thus Memory-leak can be corrected with the help of CRT libraries.

Reference: https://msdn.microsoft.com/en-us/library/x98tx3cf.aspx

Learning outcomes

- Porting functions from python to C.
- Learning how the system information like connection information is stored in OS like Linux.
- Learning about C programming in windows platform using windows specific functions and header files.
- Creating libraries in C and using them with different applications.
- Writing tests for testing the functions or programs using pytest.
- Using CFFI for using C functions (which has better performance) in python (which is easy to code) thus increasing the performance and readability of the overall Program.