САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П.КОРОЛЕВА

Отчет по дополнительному заданию по дисциплине «Операционные системы»:

Студент: Рахметов И.П.
Группа: 6302-090301D

Преподаватель:
Востокин С.В.

Самара 2022

Введение

Lua — мультипарадигмальный интерпретируемый язык программирования, отличительной чертой которого является минималистичность: стандартная библиотека языка предоставляет лишь базовый функционал, которой, впрочем, достаточен для создания небольших и средних проектов. Преимуществом данного подхода является малый размер ядра языка, позволяющий встраивать его в другие, более крупные проекты для написания скриптовых расширений для этих проектов. Недостатком данного подхода является общая ограниченность функционала, препятствующая его использованию как самостоятельного языка разработки.

Постановка задачи

Одной из множества отсутствующих возможностей Lua является функция снимка экрана. Целью данной работы является написание расширения языка, предоставляющего данный функционал. Расширение написано на языке C с использованием WinAPI. Lua предоставляет удобный API для взаимодействия с динамически подключаемыми библиотеками, в том числе для вызова функций и работы с C-структурами.

Листинг программы

Файл screenshot.c:

```
#define _UNICODE


#include <tchar.h>
#include <windows.h>

static void WINAPI screenshot_part(WCHAR* wPath, int x1, int y1, int x2, int y2)
{
  int w = x2 - x1;
  int h = y2 - y1;

  BITMAPFILEHEADER bitmap_file_header;
```

```
  BITMAPINFOHEADER bitmap_info_header;
  BITMAPINFO bitmap_info;

  ZeroMemory(&bitmap_file_header, sizeof(BITMAPFILEHEADER));
  ZeroMemory(&bitmap_info_header, sizeof(BITMAPINFOHEADER));
  ZeroMemory(&bitmap_info, sizeof(BITMAPINFO));

  bitmap_file_header.bfType = (WORD)('B' | ('M' << 8));
          bitmap_file_header.bfOffBits      =      sizeof(BITMAPFILEHEADER)      +
sizeof(BITMAPINFOHEADER);
  bitmap_info_header.biSize = sizeof(BITMAPINFOHEADER);
  bitmap_info_header.biBitCount = 24;
  bitmap_info_header.biCompression = BI_RGB;
  bitmap_info_header.biPlanes = 1;
  bitmap_info_header.biWidth = w;
  bitmap_info_header.biHeight = h;
  bitmap_info.bmiHeader = bitmap_info_header;
  DWORD bits_count = (((24 * w + 31) & ~31) / 8) * h;

  HDC screen_context = GetDC(NULL);
  HDC bitmap_context = CreateCompatibleDC(screen_context);
  BYTE* bits = NULL;
  HBITMAP bitmap = CreateDIBSection(screen_context, &bitmap_info, DIB_RGB_COLORS,
                       (void**)&bits, NULL, 0);
  HGDIOBJ old_gdi_object = SelectObject(bitmap_context, bitmap);

  BitBlt(bitmap_context, 0, 0, w, h, screen_context, x1, y1, SRCCOPY);
  ReleaseDC(NULL, screen_context);
  SelectObject(bitmap_context, old_gdi_object); // NOTE: Is necessary. Ask why.
  DeleteDC(bitmap_context);

  HANDLE file = CreateFileW(wPath,
                GENERIC_WRITE | GENERIC_READ,
                0,
                NULL,
                CREATE_ALWAYS,
                FILE_ATTRIBUTE_NORMAL,
                NULL);
  WriteFile(file, &bitmap_file_header, sizeof(BITMAPFILEHEADER), NULL, NULL);
  WriteFile(file, &bitmap_info_header, sizeof(BITMAPINFOHEADER), NULL, NULL);
  WriteFile(file, bits, bits_count, NULL, NULL);

  CloseHandle(file);
  DeleteObject(bitmap);
}
```

```
static void WINAPI screenshot_all(WCHAR* wPath)
{
   int x1 = GetSystemMetrics(SM_XVIRTUALSCREEN);
   int y1 = GetSystemMetrics(SM_YVIRTUALSCREEN);
   int x2 = GetSystemMetrics(SM_CXVIRTUALSCREEN);
   int y2 = GetSystemMetrics(SM_CYVIRTUALSCREEN);
   screenshot_part(wPath, x1, y1, x2, y2);
}
```

Файл lwscreenshot.c:

```
#include "lauxlib.h"
#include "lua.h"
#include "lualib.h"
#include "screenshot.c"
#include <windows.h>

static int Lscreenshot(lua_State* L)
{
   int n = lua_gettop(L);
   if (n < 1 || !lua_isstring(L, 1))
   {
      lua_pushliteral(L, "usage 'screenshot(path_to_bmp)' or "
                  "'screenshot(path_to_bmp,x1,y1,x2,y2)'");
      lua_error(L);
   }
   else if (n > 1 && n != 5)
   {
      lua_pushliteral(L, "usage 'screenshot(path_to_bmp,x1,y1,x2,y2)'");
      lua_error(L);
   }
   else if (n == 5)
   {
      for (int i = 2; i <= 5; i++)
         if (!lua_isnumber(L, i))
         {
            lua_pushliteral(L, "usage 'screenshot(path_to_bmp,x1,y1,x2,y2)'");
            lua_error(L);
         }

      if (lua_tonumber(L, 2) >= lua_tonumber(L, 4))
      {
         lua_pushliteral(L, "x2 must be > x1");
         lua_error(L);
      }
```

```
    if (lua_tonumber(L, 3) >= lua_tonumber(L, 5))
    {
      lua_pushliteral(L, "y2 must be > y1");
      lua_error(L);
    }
  }

  const char* path = lua_tostring(L, 1);
  int path_len = strlen(path) + 1;
  int wPath_len = MultiByteToWideChar(CP_UTF8, 0, path, path_len, NULL, 0);
  WCHAR* wPath[wPath_len];
  MultiByteToWideChar(CP_UTF8, 0, path, path_len, wPath, wPath_len);

  if (n == 1)
    screenshot_all(wPath);
  else
    screenshot_part(wPath,
              lua_tonumber(L, 2),
              lua_tonumber(L, 3),
              lua_tonumber(L, 4),
              lua_tonumber(L, 5));
  return 0;
}

static const struct luaL_Reg R[] = {
  {"screenshot", Lscreenshot},
  {        NULL,        NULL}
};

LUALIB_API int luaopen_wscreenshot(lua_State* L)
{
  luaL_newlib(L, R);
  return 1;
}
```

Компиляция осуществляется командой: gcc -Wall -Wextra -O2 -o wscreenshot.dll -fPIC -shared -I<полный путь к директории Lua>/include lwscreenshot.c screenshot.c -lgdi32 <полный путь к директории Lua>/bin/lua54.dll

Результат работы программы

Для использования библиотеки необходимо поместить DLL файл либо в директорию проекта, либо в директорию с установленной Lua, затем вызвать эту библиотеку с помощью функции *require*:

local wscr = require 'wscreenshot'.

Результатом выполнения этой команды станет создание доступной по указанному имени (в данном примере *wscr*) хэш-таблицы, содержащей по строковому ключу *screenshot* ссылку на C-функцию, которая выполняет снимок экрана и сохраняет его в файл, путь к которому является первым и обязательным аргументом функции:

wscr.screenshot('снимок.bmp')

По умолчанию, функция делает снимок всего экрана, опицонально допускается указать x/y координаты левой верхней и правой нижней области экрана, которую требуется захватить. Например:

wscr.screenshot('снимок.bmp', 100, 50, 900, 500)

Результатом выполнения данных команд является снимок всего экрана или его области, сохраненный по указанному пути в формате bmp.