WORKSHOP LUA 2.1

LUA x C++

Stack

- Envoyer / Recevoir des données à/de Lua
- Le sommet de la stack est à -1 (ou lua_gettop)
- Le bas de la stack est à 1

Création du contexte

- `lua_State *luaL_newstate (void);`
- `void luaL_openlibs (lua_State *L);`

Execution fichier / ligne Lua

- `int luaL_dofile (lua_State *L, const char *filename);`
  - `luaL_loadfile(L, filename);`
  - `lua_pcall(L, 0, LUA_MULTRET, 0);`

- `int luaL_dostring (lua_State *L, const char *str);`
  - `luaL_loadstring(L, str);`
  - `lua_pcall(L, 0, LUA_MULTRET, 0);`

Récupéra-tion

- `int lua_type (lua_State *L, int idx);`
  - `LUA_TNIL`, `LUA_TNUMBER`, `LUA_TBOOLEAN`, `LUA_TSTRING`, `LUA_TTABLE`, `LUA_TFUNCTION`, `LUA_TUSERDATA`, `LUA_TTHREAD`, and `LUA_TLIGHTUSERDATA`.
- `int lua_is* (lua_State *L, int idx);`
- `T lua_to* (lua_State *L, int idx, ...);`

==

- `T luaL_check* (lua_State *L, int idx);`

Récupéra-tion

- `int lua_getglobal (lua_State *L, const char *name);`
- `int lua_gettable (lua_State *L, int idx);`
- `int lua_getfield (lua_State *L, int idx, const char *k);`
- `int lua_geti (lua_State *L, int idx, lua_Integer i);`

Envoi

- `void lua_setglobal (lua_State *L, const char *name);`
- `void lua_settable (lua_State *L, int idx);`
- `void lua_setfield (lua_State *L, int idx, const char *k);`
- `void lua_seti (lua_State *L, int idx, lua_Integer n);`

Manipula-tion Stack

- `void lua_pop (lua_State *L, int n);`
- `void lua_push* (lua_State *L, T value);`
- `void lua_newtable (lua_State *L);`
- `void lua_insert (lua_State *L, int idx);`
- `void lua_remove (lua_State *L, int idx);`
- `void lua_replace (lua_State *L, int idx);`
- `void lua_rotate (lua_State *L, int idx, int n);`

Appel function

- ```
  void lua_call
   (lua_State *L, int nargs,
    int nresults);
  ```

- ```
  int lua_pcall
   (lua_State *L, int nargs,
    int nresults, int msgh);
  ```

Insertion function

- typedef int (*lua_CFunction)
  (lua_State *L);
- typedef struct luaL_Reg {
  const char *name;
  lua_CFunction func;
  } luaL_Reg;

- void lua_register
  (lua_State *L, const char *name,
  lua_CFunction f);
  - lua_pushcfunction(L, f);
  - lua_setglobal(L, name);

- void luaL_newlib
  (lua_State *L, const luaL_Reg l[]);

CFunction

- typedef int (*lua_CFunction) (lua_State *L);

- Nombre argument <==> lua_gettop();
  - Premier argument ==> -n
  - Dernier argument ==> -1
- Valeur de retour <==> nombre argument push sur la stack en plus
  - Premier retour ==> -n
  - Dernier retour ==> -1

Example + Exercise

- https://github.com/pival13/WorkshopLua

- Au choix:
  - Création d'un objet List en Lua + petit main en C++

    http://www.cplusplus.com/reference/algorithm/
  - Création d'une bibliothèque Algorithme en C++ + un petit main Lua:

    http://www.cplusplus.com/reference/algorithm/
  - Création d'une bibliothèque Regex en C++ + un petit main Lua:

    http://www.cplusplus.com/reference/regex/