創好リナの

Programming Seminar

6bit

File I/O

2018.09.23

ファイル操作

- ファイルとは
- ファイル操作の基本
- Luaのファイル操作
- Luaのテーブルをファイルに保存しよう

File I/O

ファイル #とは

- 1塊のデータ群
- 入出力装置などもファイルとして扱われている
- 2つに大別すると
  - テキストファイル
  - バイナリファイル
- テキストファイルはバイナリファイルの一種と言える
- データをファイルに保存することで再利用できる

テキストファイル

- テキスト用の一定バイナリ列で書かれたファイル
- [バイナリ列:文字]のルールをエンコードという
  - 文字化けはエンコードが違う時に起きる
  - UTF-8/UTF-16/Shift-JIS/Unicode ...
- アプリや種類によって違いがないので、扱いやすい

バイナリファイル

- テキストファイル以外を言う
  - 画像/動画/音声/実行ファイル/MSOffice ...
- テキストファイルのように共通の規格があるものもある(avi/mp4/mp3/png/jpg ...)
- 基本的には専用のソフトがないと中身が解読し辛いので扱いにくい

ファイル操作 #とは

- ファイルの中身を読み込んだり書き込んだりする
  - Input / Output
- Luaでは`io`がファイル操作ライブラリ
- print()/read() などの標準入出力もファイルIOの一種と言える
  - io.write()などを使うのはそのため
- 出力先、入力先を変更することを`リダイレクト`と言う

Luaのファイル操作

Luaでファイルを開く

- 書き込み読み込みの準備
- `io.open(ファイルPATH, モード)`
  - ファイルオブジェクトを返す
  - モードは以下
    - "w": 書き込み (新規/上書き)
    - "r": 読み込みのみ
    - "a": 追加書き込み (存在してなければ新規)

LuaでファイルOutput

- Lua Vanilla

```lua
local file = io.open("/path/to/file", "w")

file:write("Output\n")  -- Fileオブジェクトのwriteメソッド
-- 改行は自分でしてね
file:close()  -- 使い終わったファイルは必ずClose!!!!
```

- ComputerCraft用の便利関数

```lua
local file = fs.open("/path/to/file", "w")
file.write("Output\n")  -- "." でよい
file.writeLine("Hello") -- 最後で勝手に改行してくれる
file.close()
```

LuaでファイルOutput

- `file#flush`
  - `write()`を実行時にはまだファイルに反映されていない(メモリ上での操作のみ)
  - `flush()`を実行するとメモリ上の変更を実際にファイルに反映される
  - `close()`時に一度`flush()`される

LuaでファイルInput

- Lua Vanilla

```lua
local file = io.open("/path/to/file", "r")
-- 1行ずつ読み込んでループ
for line in file:lines() do
    print(line)
end
file:close()

-- 1回ずつ読み込む
local line = file:read()
-- 引数なしで1行 引数で読み方指定 (linesも同じ)
print(line)
file:close()
```

LuaでファイルInput

- ComputerCraft用

```lua
local file = fs.open("/path/to/file", "r")
-- 1行読む
local line = file.readLine()

-- 全部読む
local all = file.readAll()

file.close()
```

Luaのテーブルをファイルに保存しよう

Luaのテーブルをファイルに保存

- jsonのような使い方できる
- CCに便利関数があるので使うだけ簡単!
- `textutils.serialize(obj)`
  - 引数のデータを文字列表現に変換
- `textutils.unserialize(str)`
  - `serialize()`された文字列を元のデータに変換
- Serialize(直列化)
  データを文字列など可逆可能の表現に変換する

Serialize 見本

```lua
local tbl = {a = 10, b = "aaa"}

local str = textutils.serialize(tbl)
local f = fs.open("/path/to/file", "w")
f.write(str)
f.close()

-----------------------------------

local f = fs.open("/path/to/file", "r")
local str = f.readAll()
f.close()
local tbl = textutils.unserialize(str)
-- tbl = {a = 10, b = "aaa"}
```

ファイル操作ができれば
複雑なことが一気に
できるようになります。

以上

問題

6-1

- ファイルに"Hello, World!"を出力するプログラム
- そのファイルから文字列を表示するプログラム
- 以上2つのプログラムを作りなさい

6-2

- ファイルにブロック名のリストを作る
- 眼の前のブロックがブロック名リストにあるかどうかを判別するプログラムを書きなさい

おわり