navigation
- Main Page
- Recent changes
- Random page
- Help

links
- LFS Main Site
- LFS Forum
- LFS World

search
Search LFS Manual
Go | Search

toolbox
- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link
- Page information

LFS Programming

Introduction

By its nature LFS is a very programmer friendly game and, while customisation of the game engine itself is frowned upon, there are many other avenues available for budding addon writers to explore. Whether it's parsing some esoteric file format, writing your own server system, or just grabbing your friend's hotlaps from LFSWorld, LFS has something to keep even the most adventurous hackers busy!

The following category attempts to explain and document the many various programmer topics related to LFS, in hopefully a clear and useful way. Our job of course is not to teach you how to program, there are many, many books about that, but to show you the myriad possibilities for customising LFS which are right at your fingertips.

Programming Categories

InSim

Main article: InSim

InSim is a protocol which allows an external program to communicate with Live for Speed. It allows you to create a socket connection with the game and to send and receive packets of data. The InSim protocol describes how each of these packets is formatted, and any programming language which can create a network connection and send and receive strings of binary data can interface with it.

InSim Relay

With InSim Relay you can connect to any subscribed host and share a subset of InSim packets. This is mainly used in the creation of spectator style apps, such as LFSRemote.

OutSim / OutGauge

Similar to InSim, OutSim and OutGauge also allow you to create a socket connection from an external program, but specifically for the support of motion simulators and external dashboards.

LFSWorld stats

Through the use of LFSWorld stats, addon programmers are able to query the LFSWorld Web site for a whole host of information about the racers who play the game and the servers they play on.

Mods REST API

With the release of mods in 2021, there wasn't a good way to fetch information about mods. Victor therefore introduced the REST API where users can fetch nearly anything imaginable from mods. There was a LFS Forum discussion about this.

File Formats

Many of LFS's unique file formats have been documented, both officially by the development team, and unofficially by enthusiastic hackers with hex-editors.

General Programming Information

Data-Types

The LFS game client itself is written in C++ and many of data-types referenced throughout the documentation are named in accordance with that language. The following table represents a breakdown of what each data-type means and what sort of value it holds.

Data-Types

LFS	Data Type	PHP Pack Type	Description
char	1 byte signed integer	a or C	An alphanumeric character (or a number between -128 to 127)
byte	1 byte unsigned integer	C	A number between 0 and 255
word	2 byte unsigned integer	v	A number between 0 and 65,535
short	2 byte signed integer	s	A number between -32,768 and +32,767
unsigned	4 byte unsigned integer	V	A number between 0 and +4,294,967,295
int	4 byte signed integer	l	A number between -2,147,483,648 and +2,147,483,647
float	4 byte floating point number	f	A number with a decimal point

In C strings are not first-class data-types, merely arrays of characters, so you will often see strings denoted using the following syntax.

```
char[16] <variable>
```

This indicates that the variable is a string of 16 characters in length, but in reality as the last character in a C-style string is always NULL, it would only be able to hold a value of 15 characters. For php in this case, you would use the 'a' as the pack format code, with a 16 following after it's declaration.

For example:

```
pack('a16', $variable);
```

LFS Strings and Escape Codes

Strings in LFS are C-style strings, meaning that they end with a NULL character, or are often padded with NULL characters. Most other high-level languages have done away with this limitation, so it's important to strip any NULL characters from the end of a string before using it.

LFS handles strings in its own way, it uses Windows codepages sepearared by special escape characters.

. The following table represents an attempt to document each of these codepages. In reality it's not 100% accurate, in practice it should give you reasonable results for most purposes.

Codepages

Code	Name	Codepage
^L	Latin 1	CP1252
^G	Greek	CP1253
^C	Cyrillic	CP1251
^J	Japanese	CP932
^E	Central Europe	CP1250
^T	Turkish	CP1254
^B	Baltic	CP1257
^H	Traditional Chinese	CP950
^S	Simplified Chinese	CP936
^K	Korean	CP949

In addition there are several other escape codes which denote special characters in text.

Escape Codes

Code	Character
^v	|
^a	*
^c	:
^d	\
^s	/
^q	?
^t	"
^l	<
^r	>
^h	#
^^	^

And finally the colour of text is also denoted using escape codes, a table of which can be found below.

Colours

Code	Colour
^0	Black
^1	Red
^2	Light green
^3	Yellow
^4	Blue
^5	Purple
^6	Light blue
^7	White
^8	Dark green (default)
^9	Original text colour and codepage.

Categories: Guides | Programming