Table functions

WEIDU’s native tools for editing 2DA files basically assume that they are an unlabelled and irregular grid; many are, and on those assumptions the relatively low-level WEIDU approach can’t readily be improved on.

However, a large fraction of 2da files instead have a standard form: the columns are labelled by headers (in the first row after the “2da v1.0 \*\*\*\*”) bit, and the rows likewise (usually but not always in the first column), and the engine cares about entries according to what their row and column headers are, not their physical position in the file.

For an example, look at ABCLASRQ (which determines the class requirements). The rows are labelled by kit/class names; the columns are labelled by ability scores. In editing this file, you don’t care per se about editing row 5, column 2; you care about editing the “BARD, MIN\_DEX” entry.

The functions in lib\_table address 2da files on this basis. (Note that they have no way of checking whether a given 2da file *does* have this format; that’s down to you!) They also deal with tables (like the luaXX.2da files) where the rows each have a unique number.

The innards of the **clone\_row** function is shamelessly stolen from CamDawg’s cd\_ee\_kit\_appends function.

Patch functions

**append\_numbered\_row (*str* arguments -> *int* value)**

Inserts [arguments] as a new 2da row, at the bottom, with a number header equal to the lowest unused such header. (If the row headers in the file are not all integers, fails.) Returns the number header.

**delete\_table\_column (*int* colnum, *str* column -> *null*)**

Deletes the column with “column” as a header, or the column with number “colnum” (counting from zero) if “column” is unspecified. Undefined if the column does not exist or occurs more than once.

**delete\_table\_row (*str* row, [*int* lookup\_column] -> *null*)**

Deletes any rows in the table with [row] as a header. If lookup\_column is set to a nonzero integer, the “row” label is looked up in that column rather than the zeroth column. (See KITLIST for an example of where this is useful).

**get\_table\_column\_coordinate (*str* column -> *int* colnum)**

Returns the number (counting from one) of the column with header “colnum”, or -1 if it does not exist.

**get\_table\_row (*str* row -> *int* rownum)**

Returns the number (counting from zero) of the row with header “row”, or -1 if it does not exist.

**read\_table\_entry (*str* row,column, [*int* lookup\_column, *bool* dont\_complain] -> *str* value)**

Returns the (row, column) entry in the current file, or -1 if it isn’t present. If “row” is unspecified, use the row with number “rownum” (counting from zero); likewise for “column” and “colnum” (the latter counts from one). lookup\_column functions as for **delete\_table\_row.** Unless dont\_complain is set to 1, prints a warning if the entry is not found.

**renumber\_rows ([*int* start\_at] -> *null*)**

Renumbers the rows of the file from top to bottom, starting from 1 (or from [start\_at], if set).

**write\_table\_entry (*int* rownum, colnum, *str* row, column, arguments, [*int* lookup\_column, *bool* do\_not\_insert, *str* default\_fill]->*null*)**

Writes “arguments” to the (row,column) slot in the current file. If “row” is unspecified, use the row with number “rownum” (counting from zero); likewise for “column” and “colnum”. The lookup\_column variable functions as for **delete\_table\_row.**

If the column entry is not present, the install fails. If the row entry is not present, then a new row is added at the bottom with that entry, and the row is filled in with zeroes, or with [default\_fill] if set. This automatic insertion can be disabled if do\_not\_insert is set to 1, and does not work anyway if lookup\_column is nonzero; in these cases, the install fails.

Action functions

**clone\_column (*str* clone\_from, clone\_to, file, [*bool* dont\_complain] -> *null*)**

Finds the file [file], or [file].2da if that does not exist. (Fail if neither exist.) If [clone\_from] is a column header and [clone\_to] is not, copies the entire column (or the last such column if more than one exists) and inserts it at the right as a new column, with [clone\_to] as a header. If it isn’t a column header, or if [clone\_to] is already a header, does nothing and prints a warning (suppresses the warning if dont\_complain is set).

**clone\_row (*str* clone\_from, clone\_to, file, [*bool* dont\_complain] -> *null*)**

Finds the file [file], or [file].2da if that does not exist. (Fail if neither exist.) If [clone\_from] is a row header and [clone\_to] is not, copies the entire row (or the last such row if more than one exists) and insert it at the bottom as a new row, with [clone\_to] as a header. If it isn’t a row header, or if [clone\_to] is already a header, does nothing and prints a warning (suppresses the warning if dont\_complain is set).

**delete\_table\_column (*str* file, column -> *null*)**

Finds the file [file], or [file].2da if that does not exist. (Fails if neither exist.) Deletes the [column] column as per the patch function of the same name.

**delete\_table\_row (*str* file, row -> *null*)**

Finds the file [file], or [file].2da if that does not exist. (Fails if neither exist.) Deletes the [row] row as per the patch function of the same name.

**read\_table\_entry (*str* row,column,file, [*int* lookup\_column, *bool* dont\_complain] -> *str* value)**

Returns the (row,column) entry in [file], or [file].2da if that does not exist. (Fails if neither exist.) Other behaviour, and optional arguments, work as for the patch function of the same name.

**write\_table\_entry (*str* row, column, arguments, file, [int lookup\_column, *bool* do\_not\_insert, *str* default\_fill]-> *null*)**

Writes “arguments” to the (row,column) slot of [file].2da, or [file] if file.2da doesn’t exist. (Fails if neither exists.) Other behaviour, and optional arguments, work as for the patch function of the same name.