Tyab's Manic Miners .dat processing utility.

This program allows many operations on Manic Miner map .dat files.

The following features are supported:

- scanning a map file for syntax correctness.
- fixing some map errors related to tiles, heights, crystals, and ore.
- resizing a map either larger or smaller.
- merging data from one map into another map.
- replacing the script data with data from a separate file. See the script section below for details.
- Fixing some common script errors such as invalid spaces.
- Ability to remove comments from script.
- Ability to optimize script variables and event chain names based on usage.

Simple help is shown by using the -help option.

There are three types of files used by the utility.

- A source .dat file used mainly for source merge tiles, height, crystals, ore.
- An existing destination .dat file that is both input and output. It is read in, possibly merged with data from the source file and written.
- A text file treated as a script to replace the destination script section.

For both the srcmap and outmap, those files are scanned for correctness.

All parameters may be enclosed in double quotes.

Filenames that have spaces need to be enclosed in double quotes.

If outmap does not exist, you can create it by supplying a srcmap and using -copysrc which will copy all srcmap data after fixing into outmap.

If outmap exists and you use -copysrc, outmap will be overwritten and replaced with srcmap.

You can use resize and offsets, and they will result in outmap being a different size with the srcmap data at the provided offsets.

If outmap is being created by using -copysrc, then the merge options are not valid since you are already copying over all of the map's data. However as stated above you can use the resize and offset values. This can be useful if creating a larger map from a smaller map and you want the smaller map to be centered or moved elsewhere within the larger map.

When merging tiles, the original solid rock regular walls are also copied to outmap. This is so any walls that needed them are preserved.

On maps, the origin 0,0 is the upper left corner. Positive offsets move the data towards the lower right. Negative offsets move the data towards the upper left.

After operations, the outmap automatically will have the border tiles all set to solid rock regular which is tile id 38.

On the outmap, resize is always performed prior to merge, you can combine resize and merge in the same operation.

During merge, clipping is done automatically so you don't have to worry about the srcmap being larger than outmap nor worry about offsets. Offsets during merge can be negative and works as expected.

Command Line Options:

-help	display help
-srcmap *filename*	file name of a source merge .DAT
-outmap *filename*	file name of a destination .DAT
-overwrite	allow changing existing outmap
-copysrc	outmap is recreated from srcmap, implies -overwrite
-mergeheight	merge height values from srcmap into outmap
-mergecrystal	merge crystals values from srcmap into outmap
-mergeore	merge ore values from srcmap into outmap
-mergetile	merge tile values from srcmap into outmap
-mergerect *startrow, startcol, endrow, endcol*	Merge subregion from source, from start through end.
-offsetrow *number*	row offset when merging/copying srcmap into outmap, default 0
-offsetcol *number*	col offset when merging/copying srcmap into outmap, default 0
-resizerow *number*	resize outmap rows for tiles,height,resources
-resizecol *number*	resize outmap cols for tiles,height,resources
-deftile *number*	value for invalid tiles or resize, default 1 which is a simple ground tile.
-defheight *number*	value for invalid heights or resize, default 0
-defcrystal *number*	value for invalid crystals or resize, default 0
-defore *number*	value for invalid ore or resize, default 0
-mapname *string*	levelname: value saved in outmap info section
-creator *string*	creator: value to be saved in outmap info section
-fix	fix invalid/missing tile, height, crystal, ore values. Changes associated errors to warnings.
-script *filename*	file name of script file to replace outmap's script.
-sincdirs *paths*	; set of paths to search for included scripts
-sfixspace	will fix script saved in outmap removing spaces where not allowed
-snocomment	will strip comments from script except for those that use #.
-sdefine *name=value*	define variable name to be replaced with value
-sdatefmt *string*	format string for TyabScriptDate and TyabScriptIncDate, default is "y.m.d"
-soptnames	Optimize script variable and event chain names

Options may be specified in any order.

Examples:

Scan a map for correctness:

-srcmap a.dat

Copy a map to a new filename keeping the same size.

-srcmap a.dat -outmap b.dat -copysrc

Copy a map to a new filename and resize the map to be 52 rows and 48 columns

-srcmap a.dat -outmap b.dat -copysrc -resizerow 52 -resizecol 48

Copy a map to a new map with a different size offsetting rows by 10 and columns by 9.

-srcmap a.dat -outmap b.dat -copysrc -resizerow 52 -resizecol 48 -offsetrow 10 -offsetcol 9.

Resize an existing map to be 52 rows and 48 columns.

-outmap a.dat -overwrite -rowresize 52 -colresize 48

Merge the tiles and heights from one map, into another map, offsetting by 10 rows and 9 columns

-srcmap a.dat -outmap b.dat -overwrite -mergetile -mergeheight -offsetrow 10 -offsetcol 9

Merge subregion of tiles and heights from one map, from row2, column 3 through row 4, column 5, into another map, offsetting by 10 rows and 9 columns

-srcmap a.dat -outmap b.dat -overwrite -mergerect 2,3,4,5 -mergetile -mergeheight -offsetrow 10 -offsetcol 9

Use a separate script file for outmap and give list of directories to search for it and included scripts.

-outmap b.dat -script myscript.scr -sincdirs "..;..\scripts;c:\scripts;"

Script File Processing.

Using the -script and -soutdirs options you may specify a separate script file to replace the script within outmap. The file you give will be read in and it will also include any included scripts using the list of paths you give in -soutdirs. This is similar to the PATH environment variable in how you format the list of paths to search. The operation is similar to how all compilers find included files.

To find a script file the following process is used:

- The full filename provided in -script. If not found, we break the -script into a path and filename part.
- The directory where outmap is located.
- Directories where prior scripts were found.
- The ; separated list of paths specified in -soutdirs. We try both just the filename and the full filename.

The same is applied to any included script usings its filename and any path components.

-sfixspace will remove spaces/tabs from the script where it the Manic Miner engines does not allow spaces. These spaces will no longer generate errors. Using this option allows the ability to indent event chains and other readability changes in the input source files and they will be automatically fixed to work with the Manic Miner engine.

-snocomment will strip all comments, except pragma's and those that start with #. Note that comments do not slow down the Manic Miner engine since it strips all comments prior to processing the .map file.

-sdefine will allow command line defining of name=value pairs just like #pragma define name=value

-sdatefmt will allow changing the date format used when evaluating TyabScriptDate and TyabScriptIncDate macros. The default is "y.m.d" The actual value will automatically be enclosed in double quotes, so when the define is used in the script, it can directly be used anywhere a string variable can be used. See below for the full format specification you can use for the date.

-soptnames will cause all user variable and event chain names to be optimized into shorter names. All names are sorted in order of reference, with the highest reference having the shortest name. Those event chain names that interact with the block system will not be changed, nor predefined event chain names init and tick.

Pragma comments

Pragma's are special commands used by the script pre-processor. They must start at the beginning of a line, and are specified as #pragma. If #pragma is used but it is not at the beginning of a line, it is ignored but will generate a warning.

When a #pragma comment is processed it is saved into the script as ##pragma to prevent it from being processed again by a later extraction and run. When a script is read in, all #pragma commands are processed first before any other script processing happens.

#pragma include "filename"

> Filename does not have to be quoted, but is a good idea to do so. This will allow you to include another script at this point. That entire script will be embedded and then the remainder of your script will follow. Embedded scripts may embed other scripts. An error is generated if the filename cannot be found. The -sincdir option provides the paths that are used to search for the filename in addition to the path that outmap uses.
>
> There is no limit to the depth of included scripts. To prevent duplicate or recursive includes, it does check to see if a file was already included and it will be ignored if included again. This check is based on filename only, so you cannot include two files with the same name from different directories. If a duplicate or circular include is detected, a warning is displayed.

#pragma TyabScriptDate format

The last modified time for the main -script file will be appended to this comment using the format provided. You must provide the format specification. The main use for this is for documenting the version for a script automatically.

#pragma TyabScriptIncDate format

The last modified time for the current included script file will be appended to this comment using the format provided. You must provide the format specification. The main use for this is for documenting the version for a script automatically.

The format string has the same definition when used by both by these #pragmas and the TyabScriptDate and TyabScriptIncDate predefined values. The string provided by -sdatfmt will automatically be in double quotes ready to use as a string in the script. The format string has characters in it that have meaning, and any unknown characters are directly copied to the output. The characters below may be either upper or lower case.

- **Y** Year as a four digit value
- **M** Month as a two digit value, zero filled.
- **D** Day as a two digit value, zero filled.
- **H** Hours as a two digit value, zero filled.
- **N** Minutes as a two digit value, zero filled.
- **S** Seconds as a two digit value, zero filled.
- **G** Git commit hash in short form for the file (either main script, or current included script). Git is used to check to see if the filename is part of a repository and if so, git log is used to get the commit that last modified that file.

The default string for the variable substitution is "y.m.d" which can be changed with -sdatefmt. The pragmas do not have a default and you must provide the format. For example:

```
#pragma TyabScriptIncDate vy.m.d.g
```

That will generate a line in the script similar to

```
##pragma TyabScriptIncdate vy.m.d.g v2022.11.15.abcdef
```

#pragma define *name*=*value*

This is a simple macro substitution. Occurrences of name will be replaced by value (you may also add new ones from the command line by using -sdefine) If you want name to be a string, enclose value in double quotes and that string with the quotes will be put into the script. For integer values, just use a number. Name must be unique and not match any reserved word or an error is generated. The name substitution will not modify comments – only places in the script where a variable name is valid. Because all #pragma define statements are processed before any other processing of the script, ones defined later can be referenced prior to the define. This is just like how variables work in the script engine.

Name must start with an alpha and then must only contain alpha or digits. Name will match anywhere in the script a variable name is used. This means you can even redefine the names of variables. One exception is the MM script engine does allow variables that start with a digit so those variable names cannot be replaced by this pragma. It is generally known in computer science that using a variable name that starts with a number is poor practice for those few languages that do allow it since it makes is very easy to introduce ambiguity and complicates both reading and porting.

Value must be one of the following:

- Double quoted string.
- Alphanumeric name that starts with an alpha and has nothing but alpha and digits.
- Positive floating-point number.
- Positive integer.

There are predefined #pragma defines automatically setup and can be used without you needing to define them.

- **TyabMapRows** Integer value. This is the number of rows in -outmap map.
- **TyabMapCols** Integer value. This is the number of columns in -outmap map.
- **TyabScriptDate** String value. Default is "y.m.d" for the last modified time of the input script.
- **TyabScriptDate** String value. Default is "y.m.d" for the last modified time of current included script.

To use a define in your script, enclose it inside of $(name).

While the full script syntax checking is still not operational, many common issues are automatically detected and will generate errors or warnings. Uniqueness of both variables and event chain names and detection of reserved words as either variables or event chain names. Good syntax checking on all variable declarations.

Script names used in the block system are checked for validity. These are the EventCallEvent and TriggerEventChain blocks. If a name is defined in TriggerEventChain but no script calls it, a warning is generated. If an event chain is referenced in EventCallEvent

WORK IN PROGRESS: Complete script file integrity checking….