Turbo ParTool - Short readme

By Atlas

February 8, 2026

1. Exporting a .par file using PARex

PARex is a program used to export a `.par` file from **KnightShift** into a directory of `.cpp` files, which represent each section of objects in the `.par` file.

How to use:

1. We are turning on the program.
2. Enter the name of the input file along with its format.
3. Click the enter button and wait.
4. When the console window closes, we know that the program has finished running.

The program also works in ARGC&ARGV mode. Example of use:

```
PARex.exe <par_file_name.par>
```

Quick Export:

To quickly decompile a `.par` file, use `EXPORT.ps1` or `EXPORT.bat`. To modify arguments/configuration, edit the `.bat/.ps1` file using Notepad.

Example of a .bat file:

```
PARex.exe <filename.par>
```

Example of a .ps1 file:

```
.\PARex.exe .\<filename.par>
```

2. Optional PARex configuration

The **PARex.cfg** file allows you to configure how data is exported to `.cpp` files. You can decide whether the selected values will be presented as individual bits (flags) with specific labels or in raw format. If flag mapping is disabled, the data will be saved as `uint32_t`. Disabling flagging significantly speeds up the export process and reduces the compilation time of the generated code. The following parameters are available in the configuration file:

True:

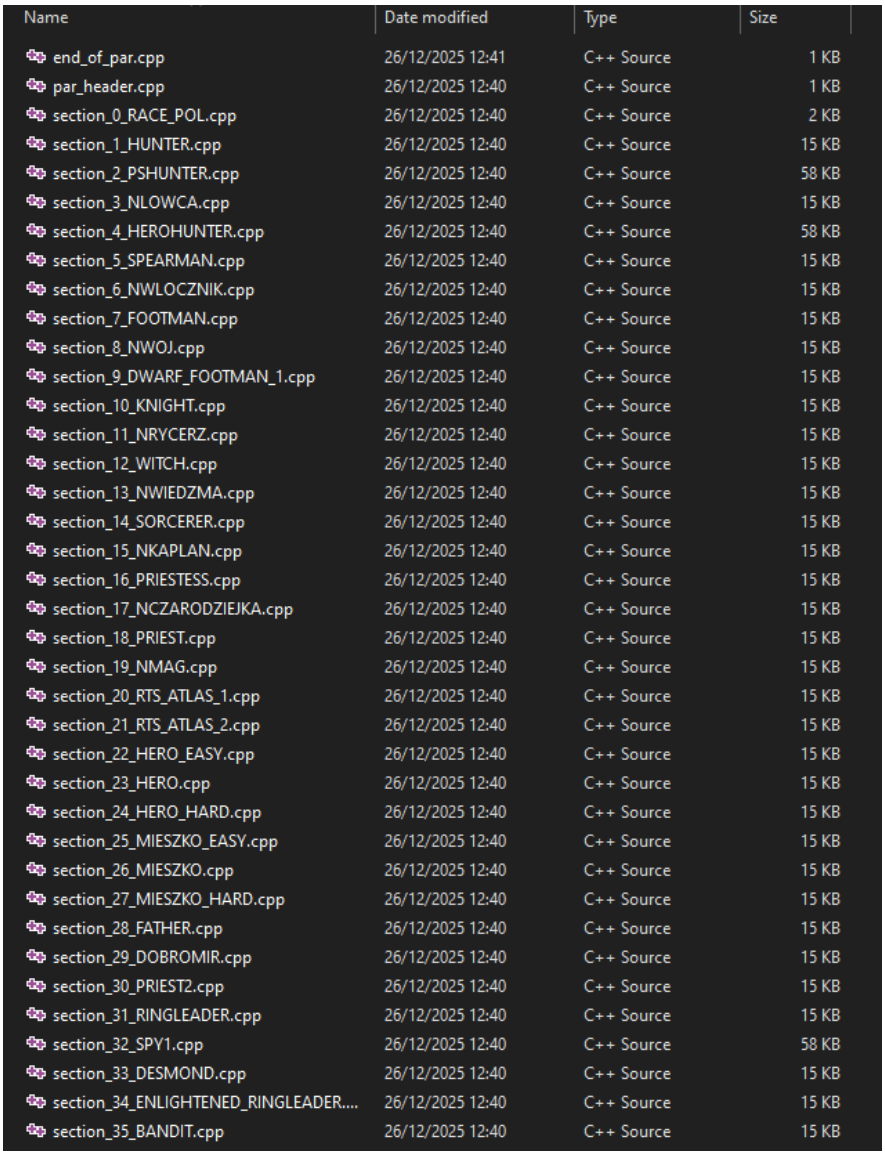
- YES/Yes/yes
- TRUE/True/true
- 1

False:

- NO/No/no
- FALSE/False/false
- 0

3. General overview of the structure of files obtained after export/decompilation

After exporting the `.par` file, we will receive a directory containing `.cpp` files representing sections with objects. The syntax of the files is in `C++`. In addition to the `.cpp` files with sections, we also receive a `section_order.txt` file, which contains the order in which the section files are compiled.



Name	Date modified	Type	Size
end_of_par.cpp	26/12/2025 12:41	C++ Source	1 KB
par_header.cpp	26/12/2025 12:40	C++ Source	1 KB
section_0_RACE_POL.cpp	26/12/2025 12:40	C++ Source	2 KB
section_1_HUNTER.cpp	26/12/2025 12:40	C++ Source	15 KB
section_2_PSHUNTER.cpp	26/12/2025 12:40	C++ Source	58 KB
section_3_NLOWCA.cpp	26/12/2025 12:40	C++ Source	15 KB
section_4_HEROHUNTER.cpp	26/12/2025 12:40	C++ Source	58 KB
section_5_SPEARMAN.cpp	26/12/2025 12:40	C++ Source	15 KB
section_6_NWLOCZNIK.cpp	26/12/2025 12:40	C++ Source	15 KB
section_7_FOOTMAN.cpp	26/12/2025 12:40	C++ Source	15 KB
section_8_NWOJ.cpp	26/12/2025 12:40	C++ Source	15 KB
section_9_DWARF_FOOTMAN_1.cpp	26/12/2025 12:40	C++ Source	15 KB
section_10_KNIGHT.cpp	26/12/2025 12:40	C++ Source	15 KB
section_11_NRYCERZ.cpp	26/12/2025 12:40	C++ Source	15 KB
section_12_WITCH.cpp	26/12/2025 12:40	C++ Source	15 KB
section_13_NWIEDZMA.cpp	26/12/2025 12:40	C++ Source	15 KB
section_14_SORCERER.cpp	26/12/2025 12:40	C++ Source	15 KB
section_15_NKAPLAN.cpp	26/12/2025 12:40	C++ Source	15 KB
section_16_PRIESTESS.cpp	26/12/2025 12:40	C++ Source	15 KB
section_17_NCZARODZIEJKA.cpp	26/12/2025 12:40	C++ Source	15 KB
section_18_PRIEST.cpp	26/12/2025 12:40	C++ Source	15 KB
section_19_NMAG.cpp	26/12/2025 12:40	C++ Source	15 KB
section_20_RTS_ATLAS_1.cpp	26/12/2025 12:40	C++ Source	15 KB
section_21_RTS_ATLAS_2.cpp	26/12/2025 12:40	C++ Source	15 KB
section_22_HERO_EASY.cpp	26/12/2025 12:40	C++ Source	15 KB
section_23_HERO.cpp	26/12/2025 12:40	C++ Source	15 KB
section_24_HERO_HARD.cpp	26/12/2025 12:40	C++ Source	15 KB
section_25_MIESZKO_EASY.cpp	26/12/2025 12:40	C++ Source	15 KB
section_26_MIESZKO.cpp	26/12/2025 12:40	C++ Source	15 KB
section_27_MIESZKO_HARD.cpp	26/12/2025 12:40	C++ Source	15 KB
section_28_FATHER.cpp	26/12/2025 12:40	C++ Source	15 KB
section_29_DOBROMIR.cpp	26/12/2025 12:40	C++ Source	15 KB
section_30_PRIEST2.cpp	26/12/2025 12:40	C++ Source	15 KB
section_31_RINGLEADER.cpp	26/12/2025 12:40	C++ Source	15 KB
section_32_SPY1.cpp	26/12/2025 12:40	C++ Source	58 KB
section_33_DESMOND.cpp	26/12/2025 12:40	C++ Source	15 KB
section_34_ENLIGHTENED_RINGLEADER....	26/12/2025 12:40	C++ Source	15 KB
section_35_BANDIT.cpp	26/12/2025 12:40	C++ Source	15 KB

Figure 1: List of .cpp files after export.

4. Editing files:

- It is a good idea to edit `.cpp` files in Geany, which presents the section in an aesthetic tree structure.
- The `section_order.txt` file contains the order of the relevant sections. When adding a new section, enter the newly added section into this file in the appropriate place.
- `count()` entries mean that during compilation, the compiler will count the number of sections/objects in `.par` (which is very convenient). If we want to enter a fixed value, we can enter it instead of `count()` and the compiler will accept it.

- To add an object, copy an existing object, paste it into the center of the section in the appropriate place, and change the data.
- The section will be compiled in the same order as the header and objects in a given section.
- Variable names can be arbitrary, but the names `number_of_objects` and `number_of_sections` are exceptions - these names are reserved for the compiler.
- The **PARim** program, i.e., the compiler, supports comments, so you can also leave them in files.

5. Compiling .cpp files into .par files using PARim

PARim is a program, or more precisely a compiler used to compile an entire directory of `.cpp` files into `.par` files.

How to use:

1. We are turning on the program.
2. Enter the name of the directory.
3. Press enter and it should be done in a moment.
4. Closing the console window signals the end of the program.

The program also works in ARGC&ARGV mode. Example of use:

```
PARim.exe <input directory name>
```

Quick Import:

To quickly compile a directory with `.cpp` files, use `IMPORT.ps1` or `IMPORT.bat`. To modify arguments/configuration, edit the `.bat/.ps1` file with Notepad.

Example of a .bat file:

```
PARim.exe <input directory name>
```

Example of a .ps1 file:

```
.\PARim.exe .\<input directory name>
```