Sandokan

- ☒ internal/external quicksort
- ☒ 32bit/64bit
- ☒ Linux/Windows
- ☒ written in C, open-source
- ☒ 3 regimes: fast/mixed/slow
- ☒ free download at www.sanmayce.com

```
E:\Sandokan_r3-++\Sandokan_vs_GNU_vs_RP>"Sandokan_QuicksortExternal_4+GB_32bit_Microsoft.exe"
Sandokan_QuicksortExternal_4+GB r.3-++, written by Kaze, using Bill Durango's Quicksort source.
Usage: Sandokan_QuicksortExternal_4+GB wordlistfile {/fast|/slow} {/ascend|/descend}
Purpose: Sandokan sorts all [CR]LF ending lines in a given file, lines can be 4++ billion i.e. more than 1<<32.
Note1: Sandokan is [sti11] in look-around stage, for superfast sorter look for FFT,
       or internal-Quicksort applied on biggest physical RAM block look for FFT.
Note2: Currently (r.3-++) the idea is to see how Quicksort behaves on all-physical, physical-external and all-external RAM.
Note3: The lines in wordlistfile must be up to 2048 chars/bytes and end with CRLF or LF i.e. maximum 2050bytes long.
Note4: The goal is to sort a file unfittable in physical memory, and with 32bit code.
Note5: When /fast is specified wordlistfile is uploaded into internal memory if it is possible.
Note6: When /fast is specified and attempt to upload the entire file is failed then new attempt is made to use 768MB for
       dynamic chunks uploading to the physical RAM block used for internal Quicksort,
       thus /fast is either all-internal or internal-external.
Note7: When /slow is specified wordlistfile is NOT uploaded into internal memory even if it is possible, thus /slow is all-external.
Note8: Not buffered I/O is just awful, like 1byte Reads/writes: fread(&FourGram[i], 1, 1, fp_in); fwrite(&FourGram[i], 1, 1, fp_outRG);.
Note9: The most cute thing about pure (r.3-++) Quicksort approach is the I/O Read-only mode through all the sorting,
       of course if all the pointers are in physical (not a part in virtual) RAM (another ugly facet: 8bytes-per-line requirement),
       the ugly thing is the huge number of random accesses i.e. seeks, to bridge this gap in next revision a new (fourth) mode
       will be available: Quicksorting all chunks (fittable in a given physical block) and then simply merging them - an
       all-sequential approach except the few seeks needed for reaching the chunk offsets.
NoteA: I cannot avoid asking myself:
       how come that sorting names of all people walked/walking on Earth (some 20 billion maybe) appeared such a difficult task,
       if Sandokan (r.3-++) is to sort (in external-internal regime) he (not it) would need 151GB (20,000,000,000x8~1GB)
       physical (preferably) RAM as a minimum. An assumption: if names are 50bytes long in average then for all-internal mode
       the needed memory jumps to 150GB+932GB, ouch.
NoteB: Any improvements are welcomed, my email: sanmayce@sanmayce.com, the idea is a free open-source sorter to become available.
E:\Sandokan_r3-++\Sandokan_vs_GNU_vs_RP>
```