Develop a Linux character device driver that will convert text data between various character encodings. The following encodings have to be supported: UTF-8, UTF-16, UTF-32, ISO8859-1, ISO8859-9.

The device nodes will be named "/dev/encode0", "/dev/encode1" etc. The number of such devices will be a module parameter. The driver will operate as follows:

- For every device, there will be an internal kernel buffer. The buffer will be persistent during the module's lifetime. Read and write operations will operate on this buffer. The buffer will have a fixed size which will be a module parameter (by default 4KB).
- The devices will be "single-open", i.e. only one process will be able to access it at a time. For example, if a process has opened "/dev/encode0" other processes will not be able to access it; but they might access "/dev/encode1". You can refer to the "Linux Device Drivers" book for an example on how to do this.
- The driver will keep a "current input encoding" and a "current output encoding". Any data written to any of the devices is assumed to be in the current input encoding. Any data read from any of the devices will be converted to the current output encoding.
- It is recommended that you use one of the UTF encodings for the data in the internal kernel buffer. That means, convert the input data from the input encoding to UTF on write and convert it from UTF to the output encoding on read.
- Do NOT use any encoding conversion functions that the Linux kernel might provide. Implement your own conversion algorithms.
- There will be two "ioctl" commands for setting the current input and output encodings.
- Supply test files to test your driver.
- Also implement the "seek" system call and include examples for its use in the test file.
- Make sure to implement meaningful behaviour when an erroneous operation is requested.