CSCI 6620 / 2226 Spring, 2016

P6: BitStreams

Encryption algorithms and compression algorithms work with bits, not bytes. The bits are normally stored in binary files. Modern languages provide a text interface for programs. They also permit creation and use of binary files, but they don't provide high-level tools for dealing with single bits and packing them into bytes that can written to a file.

Our text-interface is in the form of streams. A stream is a data structure that does a lot of things for the programmer:

- Streams provide a uniform way to access all sorts of devices.
- Streams detect and report error conditions and end-of-file.
- IStreams provide input buffering. This is allows the run-time system to read an entire disk cluster and feed it to a program one field at a time. The programmer is simply not aware of when the disk operations actually happen.
- OStreams provide output buffering. This allows a program to write one number or one line at a time, and automatically sends the material to the disk when an entire block of data has been accumulated.

A Bitstream provides the same services at the level of single bits that an istream or ostream provides at the byte level.

Bit Output Streams. BitOStream is a class that wraps `ostream`. BitOStream has a class member of type ostream, and it writes one byte to its ostream whenever needed. It supplies a public interface for programs:

- Write one bit. Pack the bit into the current byte, and write it to the ostream when 8 bits have been packed in.
- Pad and flush the remaining bits that have previously been written and not yet sent to the disk.

Bit Input Streams. A BitIStream has a class member of type istream, and it reads one byte from its istream whenever needed. It supplies a public interface for programs:

- When the client program wants a bit, the BitIStream delivers the next unused bit from its byte-buffer. It must unpack that bit from the current byte return it to the caller in the form of an integer such that the input bit must is in the rightmost bit-position. When all 8 bits have been used, the BitIStream will read another byte from the istream.
- An extra byte must follow the last data byte in the file to tell how many padding bits were used. This count must be the last byte in the file.
- The BitIStream must provide a public function for detecting end-of-data because the data may end in the middle of a byte. The eof flag in the istream is not adequate to handle this complexity. The task requires a 2-byte buffer and a 2-byte lookahead because, when the physical end of file happens, the last actual data byte will be in the first byte of the buffer and the padding count in the second byte. Then the BitIStream knows where the actual data ends.
- To read a new byte from the istream, the BitIStream must first move the untouched byte from the second slot in the buffer to the first slot, then fill the second slot with a byte from the istream.