Purpose

To produce base64-encoded story files ready for in-browser play by a Javascript-based interpreter such as Parchment.

§**1. Base 64.** This encoding scheme is defined by the Internet standard RFC 1113. Broadly, the idea is to take a binary stream of bytes, break it into threes, and then convert this into a sequence of four emailable characters. To encode 24 bits in four characters, we need six bits per character, so we need $2^6 = 64$ characters in all. Since $64 = 26 + 26 + 10 + 2$, we can nearly get there with alphanumeric characters alone, adding just two others – conventionally, plus and forward-slash. That's more or less the whole thing, except that we use an equals sign to indicate incompleteness of the final triplet (which might have only 1 or 2 bytes in it).

RFC 1113 permits white space to be used freely, including in particular line breaks, but we don't avail ourselves.

```
char *RFC1113_table = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
```

§**2.** The encoding routine is as follows.

```
void encode_as_base64(char *in_filename, char *out_filename, char *top, char *tail) {
    FILE *IN = fopen(in_filename, "rb");
    if (IN == NULL)
        fatal_fs("can't open story file for base-64 encoding", in_filename);
    FILE *OUT = fopen(out_filename, "w");                                    a text file, not binary
    if (OUT == NULL)
        fatal_fs("can't open base-64 encoded story file for output", out_filename);
    if (top) fprintf(OUT, "%s", top);
    while (TRUE) {
        int triplet[3], triplet_size = 0;
        ⟨Read the triplet of binary bytes, storing 0 to 3 in the size read 3⟩;
        if (triplet_size == 0) break;
        int quartet[4];
        ⟨Convert triplet to a quartet 4⟩;
        int i; for (i=0; i<4; i++) fputc(RFC1113_table[quartet[i]], OUT);
        if (triplet_size < 3) break;
    }
    if (tail) fprintf(OUT, "%s", tail);
    fclose(IN); fclose(OUT);
}
```

The function encode_as_base64 is called from 3/rel.

§**3.** If the file ends in mid-triplet, we pad out with zeros.

⟨Read the triplet of binary bytes, storing 0 to 3 in the size read 3⟩ ≡

```
triplet[0] = fgetc(IN);
if (triplet[0] != EOF) {
    triplet_size++;
    triplet[1] = fgetc(IN);
    if (triplet[1] != EOF) {
        triplet_size++;
        triplet[2] = fgetc(IN);
        if (triplet[2] != EOF)
            triplet_size++;
    }
}
int i; for (i=triplet_size; i<3; i++) triplet[i] = 0;
```

This code is used in §2.

§**4.**

⟨Convert triplet to a quartet 4⟩ ≡

```
int i; for (i=0; i<4; i++) quartet[i] = 0;
quartet[0] += (triplet[0] & 0xFC) >> 2;
quartet[1] += (triplet[0] & 0x03) << 4;
quartet[1] += (triplet[1] & 0xF0) >> 4;
quartet[2] += (triplet[1] & 0x0F) << 2;
quartet[2] += (triplet[2] & 0xC0) >> 6;
quartet[3] += (triplet[2] & 0x3F) << 0;
switch (triplet_size) {
    case 1: quartet[2] = 64; quartet[3] = 64; break;
    case 2: quartet[3] = 64; break;
}
```

This code is used in §2.