Templates 3/templ

Purpose

To manage templates for website generation.

3/templ.§1-4 Defining template paths; §5-6 Searching for template files

Definitions

¶1. Template paths define, in order of priority, where to look for templates.

```
typedef struct template_path {
    char template_repository[MAX_FILENAME_LENGTH];        pathname of folder of repository
    MEMORY_MANAGEMENT
} template_path;
```

The structure template_path is private to this section.

¶2. Templates are the things themselves.

```
typedef struct template {
    char template_name[MAX_FILENAME_LENGTH];              e.g., "Standard"
    struct template_path *template_location;
    char latest_use[MAX_FILENAME_LENGTH];                 filename most recently sought from it
    MEMORY_MANAGEMENT
} template;
```

The structure template is private to this section.

§1. Defining template paths.

The following implements the Blurb command "template path".

```
int no_template_paths = 0;
void new_template_path(char *pathname) {
    template_path *tp = CREATE(template_path);
    strcpy(tp->template_repository, pathname);
    if (trace_mode)
        printf("! Template search path %d: <%s>\n", ++no_template_paths, pathname);
}
```

The function new_template_path is called from 1/blurb.

§**2.** The following searches for a named file in a named template, returning the template path which holds the template if it exists. This might look a pretty odd thing to do – weren't we looking the file itself? But the answer is that `seek_file_in_template_paths` is really used to detect the presence of templates, not of files.

```
template_path *seek_file_in_template_paths(char *name, char *leafname) {
    template_path *tp;
    LOOP_OVER(tp, template_path) {
        char possible[MAX_FILENAME_LENGTH];
        sprintf(possible, "%s%c%s%c%s",
            tp->template_repository, SEP_CHAR, name, SEP_CHAR, leafname);
        if (file_exists(possible)) return tp;
    }
    return NULL;
}
```

The function seek_file_in_template_paths is.

§**3.** And this is where that happens. Suppose we need to locate the template "Molybdenum". We ought to do this by looking for a directory of that name among the template paths, but searching for directories is a little tricky to do in ANSI C in a way which will work on all platforms. So instead we look for any of the four files which compulsorily ought to exist (or the one which does in the case of an interpreter; those look rather like website templates).

```
template *find_template(char *name) {
    template *t;
    ⟨Is this a template we already know? 4⟩;
    template_path *tp = seek_file_in_template_paths(name, "index.html");
    if (tp == NULL) tp = seek_file_in_template_paths(name, "source.html");
    if (tp == NULL) tp = seek_file_in_template_paths(name, "style.css");
    if (tp == NULL) tp = seek_file_in_template_paths(name, "(extras).txt");
    if (tp == NULL) tp = seek_file_in_template_paths(name, "(manifest).txt");
    if (tp) {
        t = CREATE(template);
        strcpy(t->template_name, name);
        t->template_location = tp;
        return t;
    }
    return NULL;
}
```

The function find_template is.

§**4.** It reduces pointless file accesses to cache the results, so:

```
⟨Is this a template we already know? 4⟩ ≡
    LOOP_OVER(t, template)
        if (strcmp(name, t->template_name) == 0)
            return t;
```

This code is used in §3.

§5. Searching for template files. If we can't find the file `name` in the template specified, we try looking inside "Standard" instead (if we can find a template of that name).

```
int template_doesnt_exist = FALSE;
char *find_file_in_named_template(char *name, char *needed) {
    template *t = find_template(name), *Standard = find_template("Standard");
    if (t == NULL) {
        if (template_doesnt_exist == FALSE) {
            errorf_1s(
                "Websites and play-in-browser interpreter web pages are created "
                "using named templates. (Basic examples are built into the Inform "
                "application. You can also create your own, putting them in the "
                "'Templates' subfolder of the project's Materials folder.) Each "
                "template has a name. On this Release, I tried to use the "
                "'%s' template, but couldn't find a copy of it anywhere.", name);
        }
        template_doesnt_exist = TRUE;
    }
    char *path = try_single_template(t, needed);
    if ((path == NULL) && (Standard))
        path = try_single_template(Standard, needed);
    return path;
}
```

The function find_file_in_named_template is called from 3/rel.

§6. Where, finally:

```
char *try_single_template(template *t, char *needed) {
    if (t == NULL) return NULL;
    sprintf(t->latest_use, "%s%c%s%c%s",
        t->template_location->template_repository, SEP_CHAR, t->template_name, SEP_CHAR, needed);
    if (trace_mode) printf("! Trying <%s>\n", t->latest_use);
    if (file_exists(t->latest_use)) return t->latest_use;
    return NULL;
}
```

The function try_single_template is.