Command Line and Errors 1/cli

Purpose

To parse the command line arguments with which inweb was called, and to handle any errors it needs to issue.

Definitions

¶1. The command line options set the following variables. True/false options have `*_switch` variables, while textual ones are `*_setting`. (They also cause the two main program modes to be set: these are defined in Program Control.)

```
$analyse_structure_setting = "";
$catalogue_switch = 0;
$functions_switch = 0;
$convert_graphs_switch = 0;
$make_graphs_switch = 0;
$open_pdf_switch = -1;
$scan_switch = 0;
$tangle_setting = "";
$verbose_about_input_switch = 0;
$voids_switch = 0;
$web_setting = "";
$create_setting = "";
$only_setting = "";
$complete_PDF_leafname = "Complete.pdf";
```

`-analyse-structure`*: name of typedef struct to show usage of*
`-catalogue`*: print catalogue of sections*
`-functions`*: print catalogue of functions within sections*
`-convert-graphs`*: run* `-make-graphs` *output through 'dot'*
`-make-graphs`*: compile code to generate graphs*
`-open-pdf`*: open any woven PDF in the OS once it is made*
`-scan`*: simply show the syntactic scan of the source*
`-tangle X`*: the pathname X, if supplied*
`-verbose-about-input`*: print names of files read to stdout*
`-voids`*: print void pointer usage*
set by `-web`*: project folder relative to cwd*
set by `-create`*: name of a new project to create*
set by `-only`*: restrict a swarm to this chapter*
overridden when `-only` *is used*

¶2. In order to run, `inweb` needs to know where it is installed – this enables it to find its configuration file, the macros file, and so on. Unless told otherwise on the command line, we'll assume `inweb` is present in the current working directory.

```
$path_to_inweb_setting = "";
```

¶3. The two external tools we may need to be present, beyond standard shell tools:

```
$pdftex_configuration = 'pdftex';
$dot_utility_configuration = 'dot';
```

¶4. We count the errors in order to be able to exit with a suitable exit code.

```
$no_inweb_errors = 0;
```

---

§1. Reading the command line.

```
sub make_command_line_settings {
    my $i;
    my $targets = 0;
    ARGUMENT: for ($i=0; $i<=$#ARGV; $i++) {
        my $opt = $ARGV[$i];
        my $non_switch_follows = 0;
        if (($i < $#ARGV) && (not($ARGV[$i+1] =~ m/^-/))) { $non_switch_follows = 1; }
        if ($opt =~ m/^-/) ⟨Parse this as a switch 2⟩
        else {
            if ($web_setting eq "") { $web_setting = $opt.'/'; }
            else ⟨Parse this as a target sigil 7⟩;
        }
    }
}
```

§2.

⟨Parse this as a switch 2⟩ ≡

```
    $opt =~ s/^\-\-/\-/;                                    allow a doubled-dash as equivalent to one
    if ($opt eq "-test-extensions") {
        print "(Test inweb's implementation of Inform's C extensions)\n";
        full_test_double_squares();
        exit(0);
    }
    if ($opt eq "-verbose-about-input") {
        $verbose_about_input_switch = 1; next ARGUMENT;
    }
    if ($opt eq "-at") {
        if ($non_switch_follows == 1) {
            $path_to_inweb_setting = $ARGV[$i+1]; $i++; next ARGUMENT;
        }
        inweb_fatal_error("-at must be followed by the pathname where inweb lives");
    }
    ⟨Parse analysis options 3⟩;
    ⟨Parse weaver options 4⟩;
    ⟨Parse tangler options 5⟩;
    ⟨Parse creation option 6⟩;
    inweb_fatal_error("unknown command line switch: $opt");
```

This code is used in §1.

§3.

⟨Parse analysis options 3⟩ ≡

```
    if ($opt eq "-analyse-structure") {
        if ($non_switch_follows) {
            $analyse_structure_setting = $ARGV[$i+1]; $i++;
            enter_main_mode($ANALYSE_MODE);
            next ARGUMENT;
        }
        inweb_fatal_error("-analyse-structure must be followed by a structure name");
    }
    if ($opt eq "-catalogue") {
        $catalogue_switch = 1; enter_main_mode($ANALYSE_MODE); next ARGUMENT;
    }
    if ($opt eq "-functions") {
        $functions_switch = 1; enter_main_mode($ANALYSE_MODE); next ARGUMENT;
    }
    if ($opt eq "-voids") {
        $voids_switch = 1; enter_main_mode($ANALYSE_MODE); next ARGUMENT;
    }
    if ($opt eq "-make-graphs") {
        $make_graphs_switch = 1; enter_main_mode($ANALYSE_MODE); next ARGUMENT;
    }
    if ($opt eq "-convert-graphs") {
        $make_graphs_switch = 1; $convert_graphs_switch = 1;
        enter_main_mode($ANALYSE_MODE); next ARGUMENT;
    }
    if ($opt eq "-scan") {
        $scan_switch = 1; enter_main_mode($ANALYSE_MODE); next ARGUMENT;
    }
```

This code is used in §2.

§4.

⟨Parse weaver options 4⟩ ≡

```
    if ($opt eq "-weave") {
        enter_main_mode($WEAVE_MODE); next ARGUMENT;
    }
    if ($opt eq "-open") {
        $open_pdf_switch = 1; enter_main_mode($WEAVE_MODE); next ARGUMENT;
    }
    if ($opt eq "-closed") {
        $open_pdf_switch = 0; enter_main_mode($WEAVE_MODE); next ARGUMENT;
    }
    if ($opt eq "-only") {
        if ($non_switch_follows) {
            $only_setting = $ARGV[$i+1]; $i++;
            enter_main_mode($WEAVE_MODE);
            next ARGUMENT;
        }
        inweb_fatal_error("-only must be followed by a chapter number or appendix letter");
    }
```

This code is used in §2.

§**5.**

⟨Parse tangler options 5⟩ ≡
```
if ($opt eq "-tangle") {
    enter_main_mode($TANGLE_MODE); next ARGUMENT;
}
if ($opt eq "-tangle-to") {
    if ($non_switch_follows) {
        $tangle_setting = $ARGV[$i+1]; $i++;
        enter_main_mode($TANGLE_MODE); next ARGUMENT;
    }
    inweb_fatal_error("-tangle-to must be followed by a filename to write");
}
```

This code is used in §2.

§**6.** The single creation option is an exception, since it doesn't act on an existing web:

⟨Parse creation option 6⟩ ≡
```
if ($opt eq "-create") {
    if ($non_switch_follows == 1) {
        $create_setting = $ARGV[$i+1]; $web_setting = $create_setting; $i++;
        enter_main_mode($CREATE_MODE);
        next ARGUMENT;
    }
    inweb_fatal_error("-create must be followed by the pathname of a web");
}
```

This code is used in §2.

§**7.** A command-line argument not starting with a hyphen, and not already soaked up by a preceding argument such as `-tangle-to`, is a target sigil such as `2/eg` or `B`. Note that appendices are lettered A to O, but that `P` means the preliminary pages.

⟨Parse this as a target sigil 7⟩ ≡
```
$targets++;
if ($targets > 1) { inweb_fatal_error("at most one target may be given"); }
$swarm_mode = $NO_SWARM;
if ($opt eq "index") {
    $swarm_mode = $SWARM_INDEX;
} elsif ($opt eq "chapters") {
    $swarm_mode = $SWARM_CHAPTERS;
} elsif ($opt eq "sections") {
    $swarm_mode = $SWARM_SECTIONS;
} elsif ($opt eq "all") {
    $sigil_of_target = "0";
} elsif ($opt =~ m/\//) {
    $sigil_of_target = $opt;
} elsif ($opt =~ m/^\d+$/) {
    $sigil_of_target = $opt;
} elsif ($opt =~ m/^[A-O]$/) {
    $sigil_of_target = $opt;
} elsif ($opt =~ m/^P$/) {
    $sigil_of_target = $opt;
} else {
```

```
        inweb_error("target not recognised: $opt");
        print "The legal targets are:\n";
        print "   all: complete web\n";
        print "   P: all preliminaries\n";
        print "   1: Chapter 1 (and so on)\n";
        print "   A: Appendix A (and so on, up to Appendix O)\n";
        print "   3/eg: section with abbreviated name \"3/eg\" (and so on)\n";
        print "   index: HTML page indexing project\n";
        print "   chapters: all individual chapters\n";
        print "   sections: all individual sections\n";
        exit(1);
    }
```

This code is used in §1.

§**8.** We can only be in a single mode at a time:

```
sub enter_main_mode {
    my $new_mode = $_[0];
    if ($web_mode == $NO_MODE) { $web_mode = $new_mode; }
    if ($web_mode != $new_mode) {
        inweb_fatal_error("can only do one at a time - weaving, tangling or analysing");
    }
}
```

§**9. The configuration file.** `indoc` has only a tiny configuration file, mainly to point it to other tools it may need to use. Note that it needs none of these for tangling, so it doesn't actually matter if the settings are wrong in such a run.

```
sub read_configuration_file {
    my $cl;
    open(CONFIG, $path_to_inweb_setting.'inweb/Materials/inweb-configuration.txt')
        or die "inweb: can't open configuration file";
    while ($cl = <CONFIG>) {
        $cl =~ m/^\s*(.*?)\s*$/; $cl = $1;
        if ($cl =~ m/^\#/) { next; }                                    skip comment lines
        if ($cl eq "") { next; }                                        skip blank lines
        if ($cl =~ m/^(\S+)\s*=\s*(.*?)$/) {
            my $setting = $1;
            my $value = $2;
            ⟨Make one of the configuration settings 10⟩;
        }
    }
    close CONFIG;
}
```

§**10.** There's very little to see here:

⟨Make one of the configuration settings 10⟩ ≡

```
    if ($setting eq "pdftex") { $pdftex_configuration = $value; next; }
    if ($setting eq "dot") { $dot_utility_configuration = $value; next; }
    if ($setting eq "open-command") { $open_command_configuration = $value; next; }
    inweb_error("inweb: bad configuration setting ($setting)");
```

This code is used in §9.

§**11. Error messages.** Ah, they kill you; or they don't.

```
sub inweb_fatal_error {
    my $message = $_[0];
    print STDERR "inweb: $message\n";
    exit(1);
}
sub inweb_error {
    my $message = $_[0];
    $no_inweb_errors++;
    print STDERR "inweb: $message\n";
}
sub inweb_error_at {
    my $message = $_[0];
    my $file = $_[1];
    my $line = $_[2];
    $no_inweb_errors++;
    print STDERR "inweb: $message\n";
    print STDERR "  (", $file, " line ", $line, ")\n";
}
sub inweb_error_at_program_line {
    my $message = $_[0];
    my $i = $_[1];
    my $sec = $line_sec[$i];
    inweb_error_at($message, $section_pathname_relative_to_web[$sec], $line_source_file_line[$i]);
}
```