NAME

init – object cache initialization and shutdown methods

SYNOPSIS

```
#include <ShoreApp.h>
static shrc Shore::init(int     &argc,
      char     *argv[],
      const char *progname = 0,// default: argv[0] used
      const char *rcfile = 0); // default: ".shoreconfig"

static shrc Shore::exit();

static shrc Shore::default_options(int    &argc,
      char     *argv[],
      const char *progname = 0,// default: argv[0]
      const char *rcfile = 0); // default: ".shoreconfig"

static shrc Shore::process_options( int &argc,
      char        *argv[],
      const char *progclass,  // type.progclass.program.option
      const char *progname,   // overrides argv[0] if set
      const char *rcfilename,  // default: no file read
      const char *usagestring, // "usage" message
      setup_options_func func, // for application options
      option_group_t **res,    // for application options
      bool       process_hv=true );

static shrc Shore::init();    // for use only with
      // Shore::process_options  or
      // Shore::default_options
```

DESCRIPTION

Init initializes the object cache and initiates a connection with the Shore Server. It must be called before attempting to being a transaction. **Init** does not begin a transaction; see **transaction(oc).**

Init also checks to see if options have been initialized. If not, the first (long) form of **init** calls **default_options,** passing the given four arguments. (Options processing is described below.)

Exit terminates the connection with the Shore server and frees all memory resources used by the object cache. A running transaction must be committed before calling this method; if a transaction is running when this method is called, it will be aborted. For many applications it is not necessary to call this method at all, as the connection with the Shore server will be severed when the application process exits. However, applications wishing to reclaim resources held by the object cache can use this method to do so. Furthermore, **init** can be called again after **exit** if the application wishes to run more transactions, although any number of transactions can be run after a single call to **init.**

Default_options initializes the Shore options and reads the option configuration file *rcfile*. If *rcfile* is not given, it uses ".shoreconfig". If *rcfile* is a relative pathname (does not start with "/"), **default_options** first searches for it in in the current directory, then in *$HOME*. Any line in the *rcfile* that matches

```
shore.client.progname.optionname: value
```

will be used to set an option value.

After reading the *rcfile* (or skipping that step if no file name is given), the command line is read for values to override any option values that are already set. The command line is passed in with *argc* and *argv*. Command-line arguments of the form

```
-optionname value
```

are recognized, processed, removed from the array *argv*, and *argc* is decremented accordingly.

Two special options are processed and removed: if *-h* is encountered, the method prints the usage information for the options and exits. If *-v* is encountered, the method prints the current values of the options and continues.

For more information on options, see **options(svas),** specifically the section on client options.

Process_options allows more sophisticated use of the options facility. For an example of how to use this function, see **process_options(oc)** or the OO7 example in examples/oo7. The arguments to **process_options** are as follows: *Argc* and *argv* describe the command line. *Progclass* allows the caller to override the program class "client" (used if you rely on **default_options**). *Progname* allows the caller to override the program name *argv[0]*. *Rcfilename* allows the caller to specify a configuration file to be read or to cause no configuration file to be read. (**Default_options** always reads a configuration file.) If it is not given, or if a null pointer is given, no file is read. *Usagestring* is printed when **process_options** discovers that any required options are not set, and it is printed along with specific information about each option when *-h* is encountered. *Func* and *res* allow the caller to use application-specific options. *Func* can *be* null, but *res* cannot. **Process_options** creates an *option_group_t.* Each layer of software installs (adds to the option group) a description of each option it wishes to use (see **process_options(oc)** for an example). *Func* is a call-back function that allows the application to install its options. Finally, using the option group, **process_options** then scans the configuration file and command line for values for the options. *Process_hv* determines whether the special processing of *-h* and *-v* are done (as described above).

The second **(short)** form **of init** is to be used only if options have already been processed by **process_options** before **init** is called.

VERSION

This manual page applies to Version 1.1 of the Shore software.

SPONSORSHIP

The Shore project is sponsored by the Advanced Research Project Agency, ARPA order number 018 (formerly 8230), monitored by the U.S. Army Research Laboratory under contract DAAB07-91-C-Q518.

COPYRIGHT

SEE ALSO

process_options(oc), transaction(oc), options(oc), and **options(svas).**