Settings.cc

The settings package allows programmers to associate strings with values – either numerical, string, or boolean. These can be set and accessed at startup or thereafter. There is functionality for reading settings from files as well as parsing command-line arguments into the settings table so if the programmer parses the command line after reading default settings files, the command line can override any setting.

By convention, settings use the form *system.setting* to allow various libraries and subsystems to maintain global settings that are distinct and safe from overwriting by other users of common names. Each time a *setting* is given a value, the settings package does its best to coerce the setting into a usable form for all supported types: int, double, bool, and string. These may be accessed using the following functions:

```
int    getSysPropertyInt(string name,int def=0);
double getSysPropertyDouble(string name,double def=0);
bool   getSysPropertyBool(string name,bool def=false);
string getSysPropertyString(string name,string def="none");
```

The default parameter specifies the value to return if the named setting has not been set.

Functions of note for setting based on user-supplied preferences:

```
void args2SysProperties(int argc, char** args);
void args2SysProperties(int argc, char** args, string prefix);
```

Parses Unix-style command-line args to settings. The “prefix” variable will prepend the supplied string to any command-line argument that does not already follow the *system.setting* convention. Command-line arguments are processed by this function if they are prefixed by two dashes.

```
void readPropertiesFromFile(const string& complete_path,
                const string& prefix);
void readPropertiesFromFile(const string& complete_path);
void readSystemProps(const string& sysname);
void readSystemPropsCWD(const string& sysname);
void readSystemPropsInDir(const string& dname,const string& sysname);
```

These functions read system properties from files. The first two assume a complete path is supplied, and will prepend *prefix* as above if settings are not qualified in the specified file. ReadSystemProps assumes that the file to be read is $(HOME)/.$(SYSNAME)_config.

Settings within the settings file should follow the following format:

system.setting = value

At times it may be desirable to have a library or subcompilation unit load its own settings file without requiring the application programmer to have to explicitly take care of it. For this purpose, a dummy class called systemPreInit is provided. The library programmer then need only create an instance of this object, supplying the system name, to attempt a system settings load at runtime:

```
settings::systemPreInit mpi("mcl");
```