Knobs

Introduction

"Knobs" is a common term for a method of controlling various parameterizable aspects of a simulation. However, the implementation and usage is far from common and usually not as complete as what is going to be presented here. This implementation is feature rich and provides natural interfaces to practically all languages that are likely to be used in a design verification environment.

Basic Usage

A comprehensive in-memory "database" of knobs is first created by collecting knob values from the command line, initialization files, environment variables, and other sources. A set of access routines is then used to retrieve values of specific knobs.

Knobs consist of name and value pairs. However, in this implementation the knob names are regular expressions, thus allowing for many knobs to be specified at one time using patterns. The notion of value has also been expanded considerably to allow for more flexible value descriptions; for example, a value should be random but fall within a specified range.

Simple Example

As a simple example. Suppose someone wanted to control the debug level of a C program. Somewhere in the code would be a request to get the debug level knob. Let's assume its name is "debug_level". The following code would achieve that:

```
#include "knobs.h"
int main(int argc, char *argv[]) {
  knobs_init(argc, argv);
  int debug_level = knobs_get_value("debug_level", 0);
}
```

In the preceding example you see a call to knobs_init which is required to initialize the knobs database. The knob value is retrieved with the call to knobs_get_value, the first argument is the name of the knob and the second argument is a default value which is returned if the knob is not found in the database.

The simple example can be compiled as follows:

```
cc -o example -I$(KNOBS_HOME) -L$(KNOBS_HOME) -lknobs knobs.c
```

With the previous program the knob can be specified on the command line as follows:

```
example +debug_level=2
```

More Complex Example

This more complex example will demonstrate some of the more powerful features. The example will contain three knobs, two controlling the debug output from two different parts of a program and a third one controlling the probability of doing something interesting.

```
#include "knobs.h"
```

```
int main(int argc, char *argv[]) {
  knobs_init(argc, argv);
  int top_debug_level = knobs_get_value("top.debug_level", 0);
  int module_debug_level = knobs_get_value("module.debug_level", 0);
  int interesting_probability = knobs_get_value("interesting_probability", 50);
}
```

We might want to turn on all debug and change the probability to a random value between 60 and 80 percent. This can be specified as follows:

```
example +*debug_level=2 +interesting_probability=60-80
```

In the previous command normal glob style pattern matching was used. You can use the more powerful extended regular expression syntax by placing an extra plus at the beginning of knob pattern as follows:

```
example ++.*debug_[a-z]+=2 +interesting_probability=60-80
```

Unfortunately the * operator is recognized by most shells and has special meaning so it will normally be necessary to escape them as follows:

```
example +\*debug_level=2 +interesting_probability=60-80
```

Specifying Knobs in Files

Knobs can be specified in a file, for many applications this will be where most of the knobs will be found.

A knob file can contain c or c++ style comments or lines that start with # are also discarded.

Apart from comments, the file contains knobs pattern/value pairs just as they would be on the command line. For example suppose we had a file example.knobs:

```
// specify all debug levels
+*debug_level=1
// allow interesting_probability to be between 10 and 90
+interesting_probability=10-90
```

This file could be loaded into the knobs database with the -f option as follows:

```
example -f example.knobs
```

Knob Order

Knobs are read in-order into a long list. When a knob value is requested the list is searched in the same order and the final value found is returned. This allows for knobs to be specified in files that are then overridden on the command line. In fact, knobs can be specified in multiple files and listed in order on the command line to provide a way of specifying debug values and ranges and then being more specific later for running more detailed experiments. In the previous case you could do this:

```
example -f example.knobs +top.debug_level=2
```

Or to further restrict the range for interesting_probability:

example -f example.knobs +interesting_probability=70-80

Overriding Knob Order

In some cases you will want to declare a value for knob within a file and not allow it to be overridden on the command line. There is a simple way to do this. If a knobs value start with a '=' then the search for knobs is terminated and the value returned immediately. Thus you could lock down the interesting_probability knob within a file by doing:

+interesting_probability==40

Then no matter what was specified after this point it would always return 40.

Knob Sources

The knobs database is built when knobs_init is called. The knobs are collected from a number of different sources which described here in the order that they are processed.

Environment Variables

All environment variables are first entered into the database as NAME=VALUE pairs.

~/.knobsrc

The file ~/.knobsrc is loaded into the knobs database. Useful if you happen to have some personal preference. However as you will see there are ways to be more project specific and I expect this source will be rarely used.

*.knobsrc Initialization Files

All files that end in .knobsrc in any of the directories along the current working directory path are read, in order starting from the top then proceeding down.

For example, if your working directory is /home/projects/project1/experiments/experiment1 the following initialization file would be searched for:

/home/*.knobsrc

/home/projects/*.knobsrc

/home/projects/project1/*.knobsrc

/home/projects/project1/experiments/*.knobsrc

/home/projects/project1/experiments/experiment1/*.knobsrc

For example, if you wanted a knob that always had the project name in it you could create a file in

/home/projects/project1/project1.knobsrc:

+PROJECT_NAME=project1

With the above file, no matter what program ran in any directory within project1, it would always get “project1” as the result for a request to get knob “PROJECT_NAME”.

This feature is very powerful and should minimize the number of times information is required in

multiple places.

Environment Variable KNOBS

Next the environment variable KNOBS is parsed as if it was included on the command line. This gives a convenient way to temporarily override stuff that was in initialization files without the need to change the actual files.

Knob Files

Files can be specified on the command line using the -f option as the following example shows:

```
-f default.knobs -f experiment1.knobs -f my_favorite_debug_flags
```

Note that -f can be used with knob files themselves, thus providing a mechanism to group knobs from multiple files together.

Command Line

All knob values start with a + character followed by a pattern and then '=' and then a value. If a value is not specified then a value of 1 is assumed when the entry is placed in the database. Some examples:

```
+request_frequency=30
+*debug=4
+print_all_transactions
+exit_early=0
```

Value Language

The actual value of a knob stored in the database is always a string. However when the value is requested it can be retrieved directly as a string, or evaluated and returned as an integer like value. The format of a value determines how this conversion to an integer occurs. There are several different types of value formats, described as follows:

```
[0-9]+ // simple decimal integer value
0x[0-9a-fA-F]* // hexadecimal value
number1-number2 // range of values
number1,number2,number3 // list of possible values
number1:weight1,number2:weight2,number3:weight3 // weighted list of values
```

Weights are summed and then the probabilities are calculated. For example

```
20:30,25:20
```

means that 20 should have a probability of occurring 30 out of 50 times (60%) and 25 should have a probability of 20 out of 50 times (40%).

These definitions can be used recursively, some examples are as follows:

```
20-80,100 // (between 20 and 80) or 100
```

0-10:20,50-60:80 // between 0 and 10 20% of the time, between 50 and 60 80% of the time

Static verses Dynamic Random Values

One issue that arises with random knob values when retrieving them is that sometimes, in fact most of the time, you want the same value returned if you ask for it repeated times. So by default the system records that values chosen and and returns the same value on subsequent calls for the same knob name. There are access functions which bypass this functionality if you truly want a different value on each call. One such routine is knobs_get_dynamic_value.

Random Seed

Some of the value formats require making random selections. The random number generator used for this can be seeded with the +seed=n option, or with the SEED environment variable. The command line +seed option takes precedence over the environment variable. If the option is not specified then the time of day is used, essentially creating a random scenario each time the program is run. The selected seed is printed out if KNOBS_DEBUG is 1 or greater. This seed will need to be respecified if the same set of knob values is expected to be repeated.

Language Interfaces

The main knobs code is implemented in generic C, as this is the lowest common denominator between all the other languages and is easiest to interface to.

C

The full C API is shown later in this section. However the key routines are:

```
int knobs_init(int argc, char **argv);

char *knobs_get_string(char *name, char *defaultValue);

long long knobs_get_value(char *name, long long defaultValue);

long long knobs_get_dynamic_value(char *name, long long defaultValue);

void knobs_set_string(char *name, char *value);

void knobs_set_value(char *name, long long value);
```

This simplified API is what is made available in the other languages.

Usage requires included the knobs.h file and linking with the libknobs.a library file. The knob_init routine needs to be called before any of the access routines are called. An example framework is as follows:

```
#include “knobs.h”

int main(int argc, char *argv[]) {

  knobs_init(argc, argv);

  int value = knobs_get_value(“name”, 0 /*default*/);

}
```

To compile the file:

```
cc -c -I $KNOBS_HOME/c example.c
```

And to link the executable:

```
cc -o example example.o -L $KNOBS_HOME/c -lknobs
```

Full C API:

```
extern int knobs_init(int argc, char **argv);

extern int knobs_exists(char *name);
extern char *knobs_get_string(char *name, char *defaultValue);
extern long long knobs_get_value(char *name, long long defaultValue);
extern long long knobs_get_dynamic_value(char *name, long long defaultValue);

extern int knobs_add(char *pattern, char *value, char *comment);
extern int knobs_load(int argc, char *argv[], char *comment);
extern int knobs_load_string(char *name, char *buffer, char *comment);
extern int knobs_load_file(char *filename);
extern int knobs_load_file_if_exists(char *filename);

extern long long knobs_eval(char *expr);

extern char **knobs_get_files();

extern void knobs_set_string(char *name, char *value);
extern void knobs_set_value(char *name, long long value);

extern void knobs_dump();
```

Python

An example of how to use the knobs in python is as follows:

```
#!/usr/local/bin/python2.3
import knobs
import sys
knobs.init(sys.argv)
knobs.set_value('abc', 4)
knobs.set_string('xyz', 'there')

print knobs.get_value('abc', 2)
print knobs.get_string('xyz', 'hi')
```

To run the preceding example you need to add the knobs module onto the python search path as follows:

```
setenv PYTHONPATH $PYTHONPATH:$KNOBS_HOME/python
```

Perl

An example of how to use the knobs in python is as follows:

```
#!/usr/bin/env perl
use knobs;
knobs::init(\@ARGV);
print knobs::get_string("hello", "warren");
```

Tcl

Verilog

```
module test;
  initial begin
    $knobs_set_value("*ab*", 2);
    $knobs_set_string("xyz", "hi");
    $display("%m.abc=%0d", $knobs_get_value("%m.abc", 0));
    $display("xyz=%0s", $knobs_get_string("xyz", "mydefaultstring"));
  end
endmodule // top
```

Vera

```
#include "knobs.vri"
program test {
  knobs_set_value("abc", 20);
  knobs_set_string("filename", "myfile.test");
  printf("abc=%d\n", knobs_get_value("abc", 1));
  printf("filename=%s\n", knobs_get_string("filename", "nofile"));
}
```

Debug Features

An environment variable KNOBS_DEBUG controls the level of debug information that is output. 1 is minimal and is the default. Currently 3 is the most verbose.

To enable debugging:

```
setenv KNOBS_DEBUG 3
```