Korn Shell 93 libshell

Finnbarr P. Murphy
(fpm@fpmurphy.com)

In a number of posts about a year ago I discussed how to develop custom builtins for Korn Shell 93 (ksh93) using *libshell* and published APIs. You can also use these same APIs to access ksh93 functionality from within a C application. This post provides three simple examples of how to do this.

The first example simply takes two numbers as arguments on the command line, multiplies the numbers and prints out the result on *stderr*.

```
#include <shell.h>
#include <nval.h>
int main(int argc, char *argv[])
{
    Namval_t *np;
    Sfdouble_t res;
    Shell_t *shp = sh_init(argc, argv, 0);
    sh_trap("((xresult=$0*$1))", 0);
    np = nv_open("xresult", shp->var_tree, 0);
    res = nv_getnum(np);
    sfprintf(sfstderr,"%Lg\n", res);
    nv_close(np);
    return(0);
}
```

Here is how to compile this example using static libraries. I built these examples in my home directory but the ksh93 build environment is at */work/ksh93*, hence the use of *-I* and *-L* options. Note the large number of AST (Advanced Software Toolkit) and ksh93 libraries you have to specify in order to resolve all the symbols.

```
gcc -o example1 -I/work/ksh93/arch/linux.i386-64/include/ast example1.c  \
     -L/work/ksh93/arch/linux.i386-64/lib -lshell -lm -ldll -lcmd  -ldl -last
```

Here is the ouput for this example:

```
$ ./example1 3 4
12
$
```

The next example breaks the text string *string* into its component parts and prints them out to *stdout*. Note that “foo\\ bar” is one word, not two words.

```
#include <shell.h>
#include <stdio.h>
#include <nval.h>
int
main(int argc, char *argv[])
{
```

For personnal use only

 Copyright 2004-2013 Finnbarr P. Murphy. All rights reserved.

```
    char tmp[512];
    char string[] ="hello world 'my name is Finnbarr' foo\\ bar";
    sprintf(tmp, "set - %s &amp;&amp; for i in \"$@\";\n do\n echo \"$i\"\ndone", string);
    Shell_t *shp = sh_init(argc, argv, 0);
    sh_trap(tmp, 0);
    return(0);
}
```

Here is the output:

```
./example2
hello
world
my name is Finnbarr
foo bar
```

The final example is similar to the previous example, except that it creates an array and then uses *nv_getval()* to access the contents of the created array and print the contents to *stderr*.

```
#include <shell.h>
#include <stdio.h>
#include <nval.h>
int
shell(int argc, char* argv[], char *str)
{
    Namval_t *np, *np_sub;
    char tmp[512];
    sprintf(tmp, "set - %s &amp;&amp; for i in \"$@\";\n do\n aname+=(\"$i\")\ndone", str)
;
    Shell_t *shp = sh_init(argc, argv, 0);
    sh_trap(tmp, 0);
    np = nv_open("aname", shp->var_tree, 0);
    nv_putsub(np, NULL, ARRAY_SCAN);
    np_sub = np;
    do {
        // copy out the arguments to wherever here.
        fprintf(stderr, "%d: subscript='%s' value='%s'\n", np_sub, nv_getsub(np_sub), nv_g
etval(np_sub));
    } while (np_sub &amp;&amp; nv_nextsub(np_sub));
    nv_close(np);
    return(0);
}
int
main(int argc, char *argv[])
{
    char string[] ="hello world 'my name is simon' foo\\ bar";
    shell(argc, argv, string);
}
```

Here is the output:

```
$ ./example1
37212336: subscript='0' value='hello'
37212336: subscript='1' value='world'
37212336: subscript='2' value='my name is simon'
37212336: subscript='3' value='foo bar'
$
```

For personal use only

 Copyright 2004-2013 Finnbarr P. Murphy. All rights reserved.

As you can see, running shell script snippets from within a C application is not difficult. For more information on these APIs read the ksh93 *shell(3)* and *nval(3)* man pages.

For personnal use only

 Copyright 2004-2013 Finnbarr P. Murphy. All rights reserved.