# Playing with the KSH93 Version String

**Finnbarr P. Murphy**
*(fpm@fpmurphy.com)*

Korn Shell 93 (*ksh93*) is somewhat unusual in that it has a non-intuitive release versioning scheme, and it's version string contains more than simply a version number or date.

There are a number of ways of obtaining version information including the following:

```
$ echo ${.sh.version}
Version AJM 93v- 2014-12-24
$ echo $KSH_VERSION
Version AJM 93v- 2014-12-24
$ set -o vi
(Press ESC followed by Ctrl-V)
$ Version AJM 93v- 2014-12-24
```

*KSH_VERSION* was not available in *ksh93* prior to version 93t(2008) and is actually implemented as a name reference (*nameref*) to the *.sh.version* compound shell variable.

```
$ echo ${!KSH_VERSION}
.sh.version
$
```

The release version and date is defined in the *.../cmd/ksh93/include/version.h* header file as a string:

```
$ cat version.h
/***********************************************************************
 *                                                                     *
 *              This software is part of the ast package               *
 *         Copyright (c) 1982-2015 AT&T Intellectual Property          *
 *                     and is licensed under the                       *
 *                Eclipse Public License, Version 1.0                  *
 *                   by AT&T Intellectual Property                     *
 *                                                                     *
 *                A copy of the License is available at                *
 *          http://www.eclipse.org/org/documents/epl-v10.html          *
 *         (with md5 checksum b35adb5213ca9657e911e9befb180842)        *
 *                                                                     *
 *              Information and Software Systems Research              *
 *                            AT&T Research                            *
 *                           Florham Park NJ                           *
 *                                                                     *
 *                    David Korn                    *                  *
 *                                                                     *
 ***********************************************************************/
#define SH_RELEASE "93v- 2014-12-24"
```

As you can see *version.h* only contains part of the version string. So how is the full version string that you see build up?



  

From *.../cmd/ksh93/sh/init.c*:

```
...
#include        "version.h"
...
char e_version[] =
    "\n@(#)$Id: Version "
#if SHOPT_AUDIT
#define ATTRS 1
    "A"
#endif
#if SHOPT_BASH
#define ATTRS 1
    "B"
#endif
#if _AST_INTERCEPT
#define ATTRS 1
    "I"
#endif
#if SHOPT_COSHELL
#define ATTRS 1
    "J"
#endif
#if SHOPT_ACCT
#define ATTRS 1
    "L"
#endif
#if SHOPT_MULTIBYTE
#define ATTRS 1
    "M"
#endif
#if SHOPT_PFSH && _hdr_exec_attr
#define ATTRS 1
    "P"
#endif
#if SHOPT_REGRESS
#define ATTRS 1
    "R"
#endif
#if ATTRS
    " "
#endif
    SH_RELEASE " $\0\n";
```

Aa you can see from the above code snippets, the actual version string emitted by the shell contains a number of bits of useful information.

- Specific enabled compile time options such as coshell (*J*) or audit (*A*) support
- The string *93* followed by a single lowercase letter such as *u* or *v* designating a specific major release, followed by an optional minus (–) or plus (+) symbol
- Date string in the format *YYYY-MM-DD*

The major release letter is rarely incremented. A major release does not have a minus (–) or plus ( +) symbol appended to the release letter. A minus (–) appended to the release string indicates that the release is either an alpha or beta release of *ksh93* with the exact version of the alpha or beta release being designated by the date. A plus (+) appended to the release string indicates that the release is a bug fix or minor enhancement release of *ksh93* with the exact version of the release being designated by the date.

Looking at the *ksh93* source code, it appears that the release date is used to actually distinguish between various releases when necessary – not the release letter. In fact, there is specific code in *.../cmd/ksh93/sh/init.c* in the form of a *get* discipline function to simplify retrieving the release



  

date.

```
static Sfdouble_t nget_version(Namval_t *np, Namfun_t *fp) {
    const char *cp = e_version + strlen(e_version) - 10;
    int c;
    Sflong_t t = 0;
    UNUSED(fp);

    while ((c = *cp++)) {
        if (c >= '0' && c < = '9') {
t *= 10;
t += c - '0';
}
}
return (Sfdouble_t)t;
}

static const Namdisc_t SH_VERSION_disc = {0, 0, get_version, nget_version};
```

This code enables you to do the following:

```
$ echo $(( .sh.version ))
20141224
$ echo $(( KSH_VERSION ))
20141224
```

You can also do more exotic things with the release string as shown in the following trivial use of the *alarm* builtin.

```
$ alarm -r version +5
$ function version.alarm { print -n "$(echo -n "${.sh.version} "; date +%H:%S)\r"; }
$ read dummy
```

The first line specifies that the function named *version* is to be invoked every 5 seconds. The second line defines the actual function i.e. print the version string followed by the current time in HH:SS format. The third line is just a hack so that you can see the results of the previous 2 lines.

Currently there is what I would consider a significant flaw in *ksh93* regarding the name reference to *.sh.version*, i.e. the version string, The problem is that *KSH_VERSION* it is not marked readonly and can be overwritten. This means that *ksh93* scripts that reply on a particular value of *KSH_VERSION* can be fooled.

```
$ echo $KSH_VERSION
Version AJM 93v- 2014-12-24
$ KSH_VERSION=fpm
$ echo $KSH_VERSION
fpm
$ unset KSH_VERSION
$ echo $KSH_VERSION

$
```

This cannot occur with *pdksh* and derived shells where *KSH_VERSION* is a readonly shell variable.



  

Enjoy!



  