Porting Korn Shell 93 to Windows 7 SUA

Finnbarr P. Murphy

(fpm@fpmurphy.com)

Many people are unaware that the Professional and Ultimate editions of Microsoft Windows 7 come with a user-mode subsystem called Services for UNIX Applications (SUA). SUA is kind of like the poor orphaned child of the Microsoft Windows family; it is there but rarely spoken about by Microsoft.

What do I mean by a *user-mode subsystem*? Windows 7 is based on the the Windows NT architecture whose structure can be broadly divided into two parts: user-mode and kernel-mode. User-mode is made up of subsystems which pass I/O requests to the appropriate kernel-mode drivers via the I/O manager. Two types of subsystems make up the user mode layer: Environment subsystems and the Integral subsystem. The Environment subsystem is designed to run applications written for many different types of operating systems, not just Microsoft Windows. In Windows 7, there are two Environment subsystems, i.e a Win32 subsystem and a POSIX subsystem. The SUA uses the POSIX subsystem.

The version of the Korn shell that comes with Windows 7 SUA is PDKSH, i.e. Public Domain Korn Shell which is a clone of the original version of the Korn shell (*ksh88*). PDKSH is old and is not actively maintained as far I am aware. A new version of the Korn shell, Korn Shell 93 (AKA *ksh93*), was released in 1993. Since March 2000, the source code for this version has been available under an AT&T license and since 2005 under a Common Public License. *ksh93* is still actively developed by David Korn and many new features have been added over the years.

Building *ksh93* from sources requires that you have a suitable development environment available to you in the SUA. I am going to assume that this is already the case and that you have downloaded and correctly installed the Windows 7 Utilities and SDK for SUA and the relevant additional development tools from the SUA Community Tool Warehouse. I also assume that you are familiar with building executables on SUA using the GNU toolchain. BTW, I use the default *gcc-3.2* compiler for the build and not *gcc.4.2* which is available from the Tools Warehouse. In theory you could build *ksh93* with a Microsoft compiler but I have never tried to do that.

First you need to obtain the source code for the latest version of *ksh93* from AT&T Research Labs software download site. The following script will download the latest tarballs, of which there are two, required to build what is known as the *ast-ksh* package.

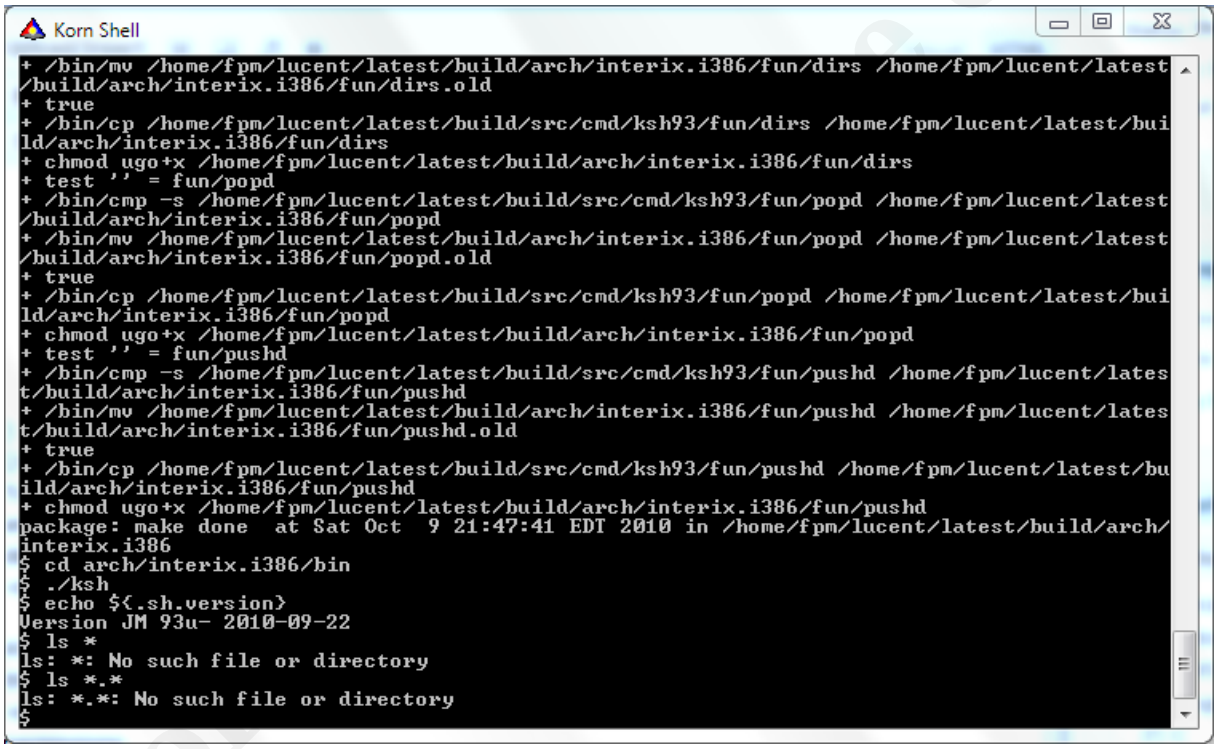
```
#!/bin/ksh
#
# FPM - 10/08/2010   (AST-KSH)
#
# Download ksh93 sources and build.
#
VERSION="2010-09-24"
wget --http-user="I accept www.opensource.org/licenses/cpl" \
 --http-passwd="." 'http://www.research.att.com/sw/download/beta/INIT.${VERSION}.tgz'
wget --http-user="I accept www.opensource.org/licenses/cpl" \
 --http-passwd="." 'http://www.research.att.com/sw/download/beta/ast-ksh.${VERSION}.tgz'
# clean up build area
[ -d build ] &amp;amp;&amp;amp; {
    echo "Deleting exiting build area ...."
    rm -rf build
```

 Copyright 2004-2019 Finnbarr P. Murphy. All rights reserved.

```
}
# populate sources
echo "Populating build area ......."
mkdir build
cd build
gunzip -c ../INIT.${VERSION}.tgz | tar -xf -
gunzip -c ../ast-ksh.${VERSION}.tgz | tar -xf -
# and build from scratch
./bin/package make
```

If you are reading this post some time after I wrote it, you may have to change the *VERSION* string to point to a later version of *ksh93*. If you do not want to use a beta version of *ksh93*, the latest stable versions can be found here. Select the *INIT ast-ksh* packages and download the source tarballs.

If all goes well your build will be successful and you will find the ksh93 binary at ../build/arch/interix.i386/bin/ksh. There is a problem with this binary however. Directory list globbing does not work correctly. See below:



BTW, for those lawyer-types who study the above image carefully and spot that a subdirectory called *lucent* is in the build path, this should not be taken to imply that Acatel-Lucent or any of its subsidiaries is the owner of, or copyright holder, of *ksh93*. The *ksh93* sources are all marked "Copyright (c) 1985-2010 AT&T Intellectual Property."

The problem is due to what information is stored in a *dirent* structure. A *dirent* structure defines a file system directory entry. Compare the SUA *dirent* structure (*/usr/include/dirent.h* below:

```
struct dirent {
    char d_name[NAME_MAX+1];
    ino_t d_ino;
    size_t d_namlen;
    long reserved[16];
};
```

 Copyright 2004-2019 Finnbarr P. Murphy. All rights reserved.

with the current GNU/Linux *direct* structure (*/usr/include/bits/dirent.h*):

```
struct dirent
  {
#ifndef __USE_FILE_OFFSET64
    __ino_t d_ino;
    __off_t d_off;
#else
    __ino64_t d_ino;
    __off64_t d_off;
#endif
    unsigned short int d_reclen;
    unsigned char d_type;
    char d_name[256];           /* We must not include limits.h! */
  };
#ifdef __USE_LARGEFILE64
struct dirent64
  {
    __ino64_t d_ino;
    __off64_t d_off;
    unsigned short int d_reclen;
    unsigned char d_type;
    char d_name[256];           /* We must not include limits.h! */
  };
#endif
```

The critical difference is the lack of a *d_type* member in the SUA *dirent* structure. This member is used to store the type of the file. A number of types are defined including:

- *DT_UNKNOWN* The file type is unknown.
- *DT_REG* The file is a regular file.
- *DT_UNKNOWN* The file is a directory.

This member is a BSD extension. On platforms where it is used, such as GNU/Linux, it corresponds to the file type bits in the *st_mode* member of the *statbuf* structure. If the value cannot be determined, *d_type* is set to *DT_UNKNOWN*.

These two macros convert between *d_type* values and *st_mode* values:

- *int IFTODT (mode_t mode)* Return the *d_type* value corresponding to *st_mode* value.
- *mode_t DTTOIF (int dtype)* Return the *st_mode* value corresponding to *d_type* value.

With this understanding, the fix for *ksh93* is trivial. Only one source code file needs to be modified and that file is *../src/lib/libast/misc/glob.c*. Add *sys/stat.h* to the list of included headers and replace the existing *gl_dirnext* function with the following:

```
static char*
gl_dirnext(glob_t* gp, void* handle)
{
        struct dirent*  dp;
        struct stat     statbuf;
        int             dt;
#define IFTODT(mode) (((mode) &amp; 0170000) >> 12)
#define DT_UNKNOWN       0
#define DT_DIR           4
#define DT_LNK          10
        while (dp = (struct dirent*)(*gp->gl_readdir)(handle))
        {
            stat(dp->d_name, &amp;statbuf);
            dt = IFTODT(statbuf.st_mode);
```

For personal use only

 Copyright 2004-2019 Finnbarr P. Murphy. All rights reserved.

```
            if (dt != DT_UNKNOWN &amp;amp;&amp;amp; dt != DT_DIR &amp;amp;&amp;amp; dt != DT_LNK)
                gp->gl_status |= GLOB_NOTDIR;
            return dp->d_name;
        }
        return 0;
}
```

This function *stat*'s every directory entry and uses the *IFTODT* macro to determine the directory entry type. It adds some overhead to *ksh93* due to the *stat* call under certain circumstances but the amount is negligible.

After you rebuild *ksh93*, you will see the the previous test we used to demonstrate the directory globing problem now executes correctly.

The next question is where to put the new executable and what to call it. I personally rename it to *ksh93* and place it in the */bin* subdirectory. Others may prefer to leave it named as *ksh* and place it in */usr/local/bin*. So long as it is on your PATH, it does not really matter. The advantage of renaming it to *ksh93* is that there is no namespace collision with the original PDKSH executable */bin/ksh*.

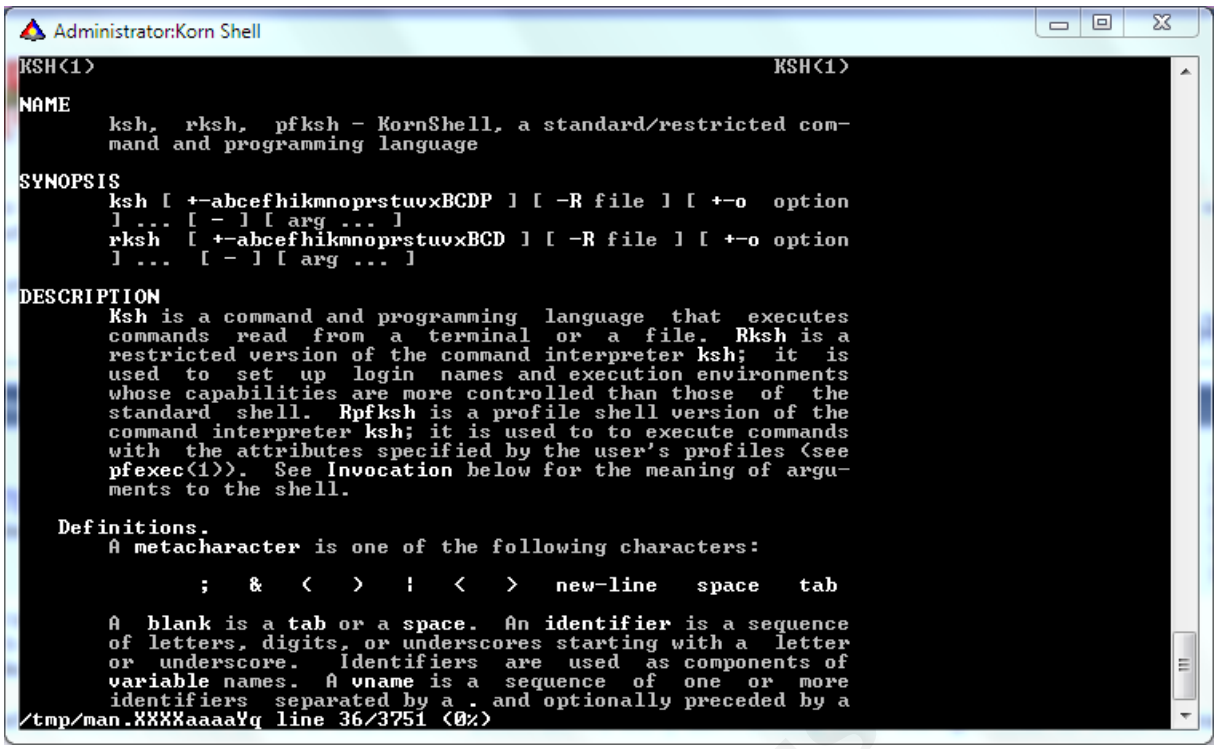
ksh93 comes with its own man page whose source is *../src/cmd/ksh93/sh.1*. Here is a shell script which makes this man page available as *ksh93(1)*.

```
#/bin/ksh
if [ `id -u` -ne '197108' ]
then
    echo "ERROR: you must be root (Administrator)"
    exit 1
fi
cp ./build/src/cmd/ksh93/sh.1  /usr/share/man/man1/ksh93.1
makewhatis
exit 0
```

Note that you must be root to run this script. Root on SUA is actually the local Administrator account (whose UID is 197108). The UID of the domain administrator is 1049076. These are fixed UIDs in SUA and cannot be safely changed.

For personal use only

 Copyright 2004-2019 Finnbarr P. Murphy. All rights reserved.



Well, that is all the information you need to get the latest version of the Korn shell build and installed in your Windows 7 SUA. You now have a seriously powerful shell available to you on Windows 7.

Enjoy!

For personal use

 Copyright 2004-2019 Finnbarr P. Murphy. All rights reserved.