JShell es un shell UNIX escrito en Java, y destinada principalmente para trabajar con los programas Java. Fue escrito para superar los problemas en el trabajo con Java en el 5mx Psion:

* eshell, la cáscara de EPOC que se puede utilizar en la 5mx, no tiene variables de entorno.
* El comando para iniciar un programa Java es bastante largo, y este problema se agrava por la falta de variables de entorno. (Por no hablar de teclado del 5mx, que es bueno para una PDA pero pequeño, no obstante.)
* Cada vez que se inicia un programa Java, la máquina virtual Java tiene que ser cargado. En la 5mx esto es que consume tiempo.
* Si desea compilar en la 5mx, javac es su única opción. Debido javac está escrito en Java, se requiere la carga de la JVM. Además, el comando para ejecutar javac es largo.
* Cada programa Java que se ejecuta de forma simultánea requiere otro JVM. No sólo hay un retraso adicional como comienza cada JVM, los requisitos de memoria de cada JVM son considerables, sobre todo en una máquina con memoria RAM de 16 megas. Este es un problema si se quiere canalizar la salida de un programa Java a otra. (Ni siquiera estoy seguro de si eshell apoya tuberías.)

Debido JShell está escrito en Java, corriendo JShell inicia una JVM. Esta es la única JVM que usted necesita. Puede utilizar JShell ejecutar otros programas Java (es decir, las clases con un método public static void main (String [])), y éstos correr dentro de JVM JShell como un hilo. Los comandos JShell son clases Java que también se ejecutan en la misma JVM. Puede ampliar el conjunto de comandos JShell.De hecho, se le anima a hacer esto y distribuir su trabajo a otros usuarios JShell.

JShell está diseñado para apoyar el desarrollo en Java. Se espera que se le va a través de editar / compilar ciclos / run / depuración. Clases de Java que no sean los comandos JShell se vuelven a cargar, con un nuevo cargador de clases, cada vez que el comando se ejecuta. Esto permite que el JShell JVM para trabajar con diferentes versiones de la clase "mismo" que permite editar y recompilar su fuente de Java.Usted no tiene que salir y reiniciar JShell para recoger las últimas versiones de sus clases de Java.

**Ejemplos**

El diseño de JShell estaba destinado a ser familiar para los usuarios de cubiertas tales como sh y csh. JShell es mucho más simple aunque algo menos potente. He aquí algunos ejemplos para darle una visión general de JShell.

Enumerar los archivos en el directorio actual, con detalles sobre cada archivo:

5> ls: l

El 5 es un "número de trabajo". Cada comando que ejecute obtiene su propio número de trabajo. El carácter predeterminado es prompt '>'. El ':' carácter se utiliza para las banderas, de forma predeterminada.El carácter de la bandera tradicional, '-', es difícil de escribir en el 5mx mientras ':' es más accesible, (al menos esto es cierto en mi 5mx; he visto otros diseños de teclado en otras 5mxs). Tanto los personajes rápidas y bandera se pueden cambiar.

Enumere las fuentes de Java en el directorio padre:

6> ls ../\*.java

Enumerar los archivos en el directorio actual, de forma recursiva, con detalles:

7> ls: LR

Puede examinar el carácter actual utilizado para banderas:

8> jshell.flag env

:

Y cambiarlo a algo más:

9> set jshell.flag -

jshell.flag = -

Entonces usted puede hacer esto, por ejemplo:

10> ls -lR

Aquí es cómo compilar todas las fuentes de Java en este directorio:

11> javac \* .java

Escriba todos los archivos .txt en el directorio actual:

12> gato \* .txt

Crea un nuevo directorio, foobar, y copiar todos los archivos .txt a ella.

13> mkdir foobar; cp \* .txt foobar

Compilar un montón de fuentes de Java en el fondo.

14> javac \* .java> errors.txt y

Una y al final de la línea hace que la línea de comandos que ingresó (incluyendo todas las declaraciones separadas por punto y coma) para ser ejecutados en el fondo. Usted recibirá un nuevo símbolo del sistema inmediatamente después de ejecutar el comando anterior, pero el lista.txt archivo no será completa hasta que finalice el comando de ejecución.

Puede ver la ejecución de los trabajos con el comando puestos de trabajo, por ejemplo,

15> Empleo

123: javac \* .java> errors.txt y

El número antes de los dos puntos es el número de "trabajo". Estos números son únicos dentro de una ejecución de JShell. Puede matar a la ejecución de los trabajos con el comando kill. Por ejemplo, el "javac \* .java> errors.txt y" comando puede ser matar así:

16> matar a 123

# Configuración y ejecución JShell

## EPOC

JShell.sis debe ser instalado en la unidad D. El jshell.bat archivo por lotes se escribe en asumir que JShell está instalado en D: \ Java \ JShell. Si usted no tiene una unidad D, es dudoso que usted tendrá suficiente memoria para el archivo classes.zip de Java, los archivos y tarro JShell tienen suficiente memoria de sobra para correr JShell eficacia. Sin embargo, si quieres probar esto, usted tendrá que modificar jshell.bat y cambiar todas las referencias a la unidad D para señalar su lugar en la unidad C.

Para ejecutar JShell, necesitará tener instalado eshell primero. Puede descargar eshell desde el sitio web de Symbian aquí.

Una vez que tenga eshell y JShell instalado, vaya al directorio D: \ Java \ JShell. Ejecute el comando "jshell". En pocos segundos, debería ver una consola con este mensaje:

0>

El primer comando tardará unos segundos para ejecutarse como los archivos de clase JShell consiguen cargado. Comandos posteriores deberían ser más rápidas.

## Windows y UNIX

Añadir jshell.jar y javacup.jar a su CLASSPATH. A continuación, ejecute jshell de la siguiente manera:

java jshell.JShell