

Using module-tcl on Windows

Table of Contents

- 1. Introduction
- 2. Getting Started
 - [How To Check For Tclsh](#)
 - [Setup](#)
 - [Slashes and spaces in pathname](#)
- 3. References
 - [How to start the cmd shell](#)
 - [Better moduleinit.cmd](#)
 - [What's with module-tcl/tcl/init Folder](#)
 - [Example: Forward slashes in pathnames](#)
 - [Example: Backward slashes in pathnames](#)
 - [Example: Mixed slashes in pathnames](#)
 - [How to use module-tcl cmd shell during startup](#)
- [Glossary](#)

Chapter 1. Introduction

Modules is a powerful package on [*Nix](#) that solves a well-defined problem - that of managing environment variables for applications and versions of applications.

This document describes the setup and usage of some minor enhancements I made to module-tcl to support the native Windows "cmd" shell. Ideally, a Win32 binary version of modules would be better. Until such time as it becomes available, the module-tcl enhancements have served me well.

In fact, this enhancement allows me to have a single set of modulefiles that work :

- a. between home machines (Win2K, Vista, Win7 and virtual machine guests) and work machines
- b. between different work machines as I change contracts/jobs
- c. between different versions of software, especially compilers on same machine
- d. provide mini documentation on the shells and tools I use

There are two features that I do not use on the cmd shell - aliases and x-resources. Without usage experience, I have left these features untested on Windows.

Finally, the goal of this document is to make the user's windows experience for Modules on Windows as productive as it has been for [*Nix](#) users. Pictures provide guidance but as Windows evolves, some minor details may differ with that of the document. Your usage experience and feedback is welcome to keep this document relevant and useful.

-- Kwee Heong Tan <tan.k.h@juno.com>.

Chapter 2. Getting Started

Table of Contents

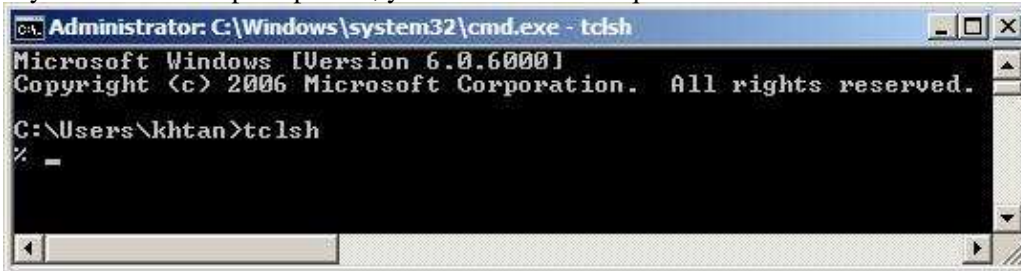
- [How To Check For Tclsh](#)
- [Setup](#)
- [Slashes and spaces in pathname](#)

How To Check For Tclsh

module-tcl requires that Tcl been installed on the machine, and that the environment variable PATH correctly set up to point to Tcl's bin. You can download ActiveState Tcl from www.activestate.com/activetcl.

Check this by opening a cmd shell and entering "tclsh". See [How To Start The Cmd Shell](#).

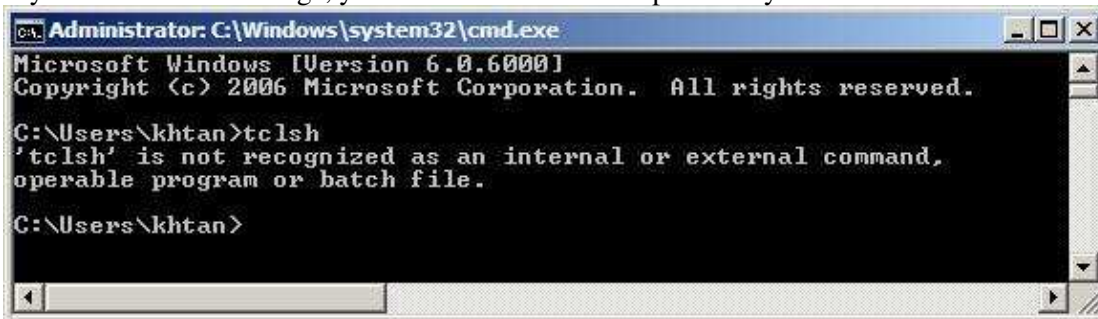
If you see the tclsh prompt "%", you have met this requirement.



```
Administrator: C:\Windows\system32\cmd.exe - tclsh
Microsoft Windows [Version 6.0.6000]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\khtan>tclsh
% -
```

If you see an error message, you do not have tclsh set up correctly.



```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.0.6000]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\khtan>tclsh
'tclsh' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\khtan>
```



Note

If both Tcl is installed and PATH is correct, but tclsh is still not found, check the tcl/bin directory. Some installations will install the specific tclsh, eg tclsh58.exe but not the generic tclsh.exe. In this case, it is simplest to just make a copy of tclsh58.exe as tclsh.exe in the said bin directory.

Setup

I. Download

Download and unzip module-tcl from <http://modules.sourceforge.org> into your favourite directory. For simplicity, use a path that has no spaces in it. Take note of the full path of the TCL folder, for example : L:/projects/git/modules-tcl/tcl.

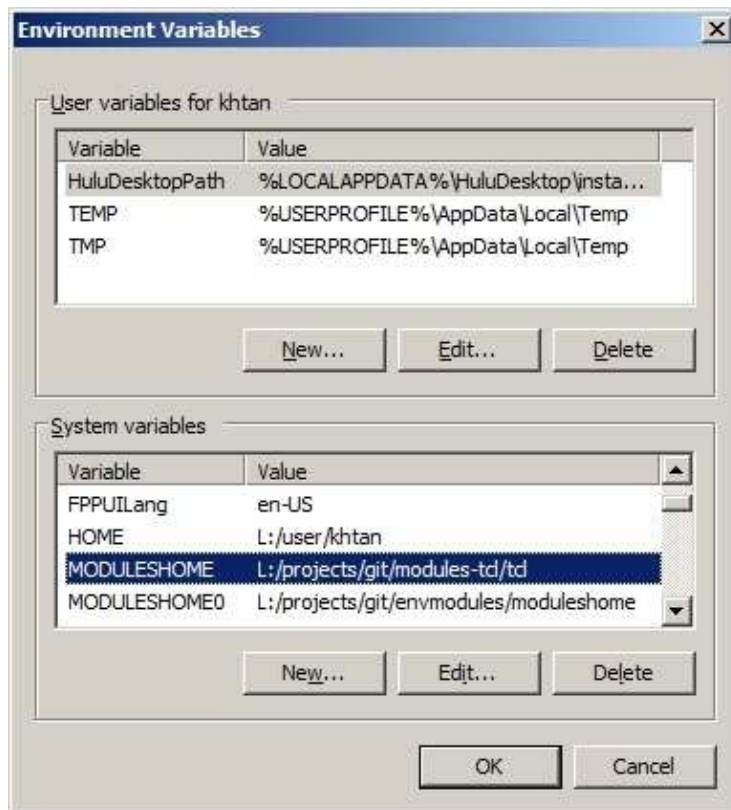


Note

In a later section [Slashes and spaces in pathname](#), we show you how to use backslashes or mixed slashes and even spaces in the pathnames. For the current setup, stick to the forward slashed, no spaces pathname first.

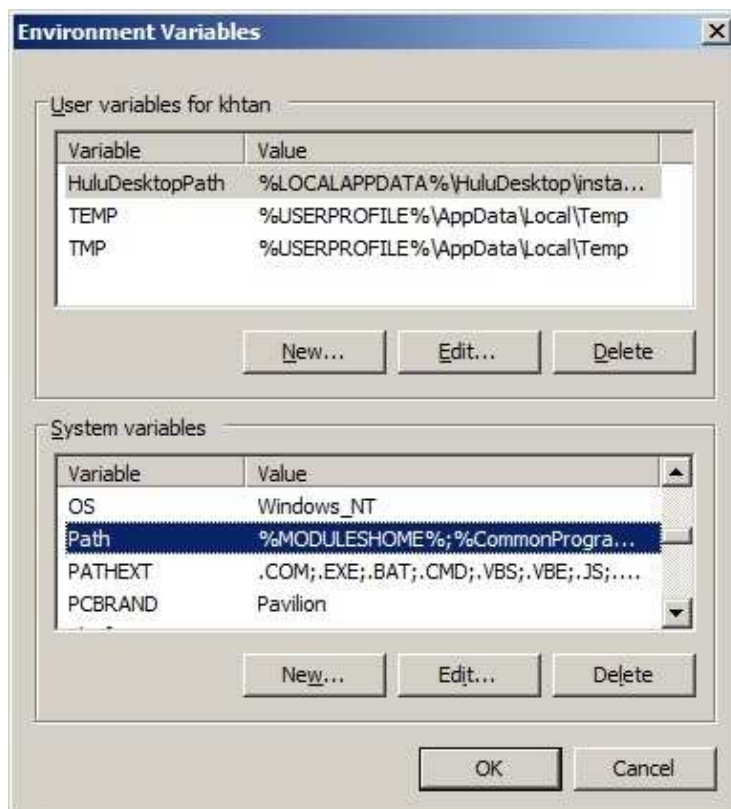
II. MODULESHOME environment variable

Create an environment variable called MODULESHOME and use the path above as its value. Check that the spelling of MODULESHOME is correct.



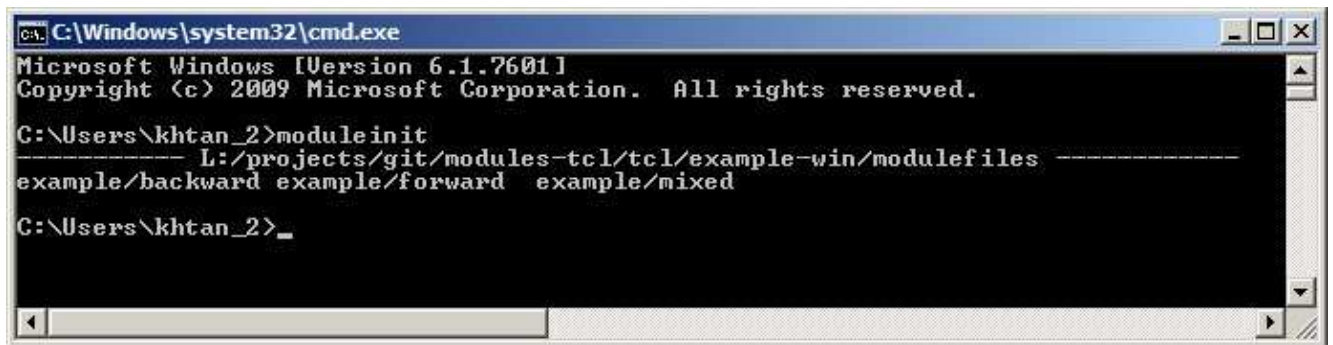
III. PATH environment variable

Add the value "%MODULESHOME%;" to the beginning of the PATH environment variable. The ";" is the separator between each path element.



IV. Check setup

Open a cmd shell ([How To Start The Cmd Shell](#)) and type "moduleinit". It is currently set up to load modulefiles from module-tcl/tcl/example-win/modulefiles. If you see a list of modulefiles similar to the picture below, the setup is working.



```

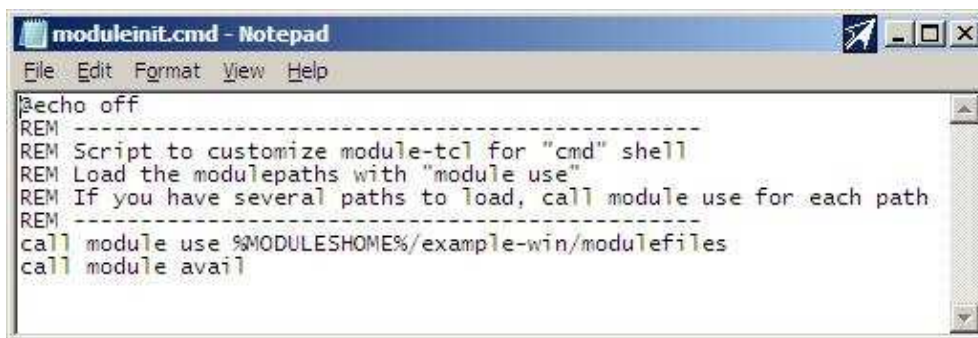
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\khtan_2>moduleinit
----- L:/projects/git/modules-tcl/tcl/example-win/modulefiles -----
example/backward example/forward example/mixed
C:\Users\khtan_2>_

```

V. Customize tcl/moduleinit.cmd

Open the file module-tcl/tcl/moduleinit.cmd and replace/add the "module use" commands to load the paths where your personal modulefiles are located. See the example below as a guide.

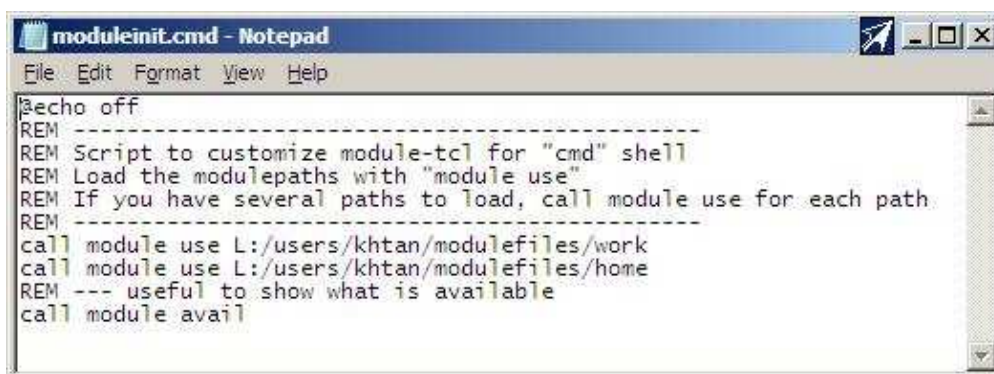


```

moduleinit.cmd - Notepad
File Edit Format View Help
@echo off
REM -----
REM Script to customize module-tcl for "cmd" shell
REM Load the modulepaths with "module use"
REM If you have several paths to load, call module use for each path
REM -----
call module use %MODULESHOME%/example-win/modulefiles
call module avail

```

The above is the original moduleinit.cmd



```

moduleinit.cmd - Notepad
File Edit Format View Help
@echo off
REM -----
REM Script to customize module-tcl for "cmd" shell
REM Load the modulepaths with "module use"
REM If you have several paths to load, call module use for each path
REM -----
call module use L:/users/khtan/modulefiles/work
call module use L:/users/khtan/modulefiles/home
REM --- useful to show what is available
call module avail

```

The above shows the new moduleinit.cmd, using 2 absolute paths to my modulefiles.

See [Better moduleinit.cmd](#) for an example of a more realistic and useful moduleinit.cmd

Slashes and spaces in pathname

It is a common myth that Windows cmd shell only supports backslashes in paths. In actuality, the cmd shell does not mind if the path separators are forward or backward slashes, and they can be mixed. It was our misfortune that Microsoft chose to use "/" as the option character, and thereafter, some utilities misinterpreted this as requiring only backslashes in paths. This non-uniformity amongst Windows commands/utilities has

reinforced this myth.

The second conundrum concerns spaces in paths. In general, *Nix does not allow this (without some work). This extra work can be used to allow module-tcl to handle spaces in the paths.

In the [Check setup step](#) of the Setup section, we invoke moduleinit to see three modulefiles in in modules-tcl/tcl/example-win/modulefiles. These are added to give you a live example of how to write pathnames with forward, backward and mixed slashes and deal with spaces in the pathnames. You can follow the step-by-step examples in each of the annotated transcripts below, by continuing with the cmd shell after moduleinit.

- ◆ [Forward slashes in pathnames](#)
- ◆ [Backward slashes in pathnames](#)
- ◆ [Mixed slashes in pathnames](#)



Note

Since the underlying engine of module-tcl is a *Nix tool, you'll find it is more natural to use forward slashes. Compare and contrast the following:

forward: "example-win/batchfiles/level 1/level 2/level 3"

backward: "example-win\\batchfiles\\level 1\\level 2\\level 3"

mixed: "example-win/batchfiles\\level 1/level 2\\level 3"



Note

Along the same line of thought, it is more natural in *Nix to avoid spaces in pathnames. You can do the same in Windows. For example, I install most of my application in c:/programs instead of C:\Program Files. Try it and see!

Chapter 3. References

Table of Contents

[How to start the cmd shell](#)

[Better moduleinit.cmd](#)

[What's with module-tcl/tcl/init Folder](#)

[Example: Forward slashes in pathnames](#)

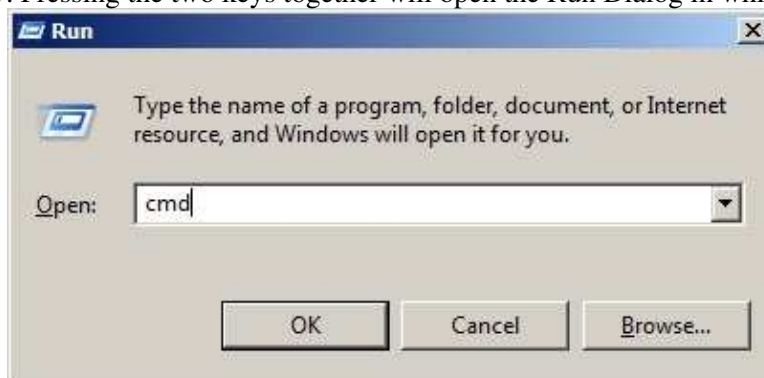
[Example: Backward slashes in pathnames](#)

[Example: Mixed slashes in pathnames](#)

[How to use module-tcl cmd shell during startup](#)

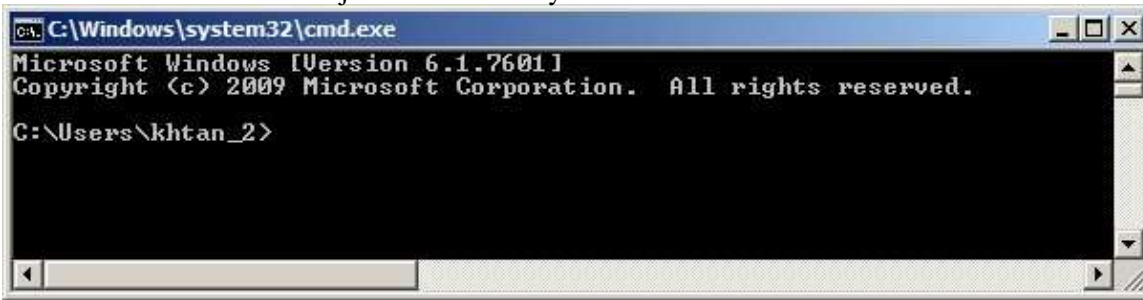
How to start the cmd shell

There are many ways to start the cmd shell. This describes a fast and convenient way - using the Windows Logo key and the "R" key. Pressing the two keys together will open the Run Dialog in which you can enter



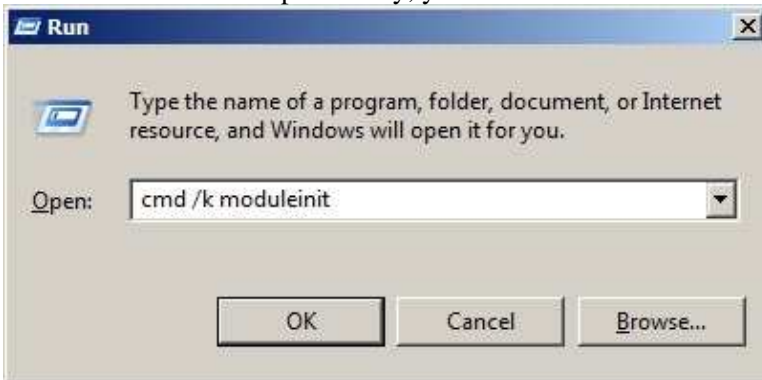
the command text "cmd".

Click on the "OK" button or just the Return key will start the cmd shell.

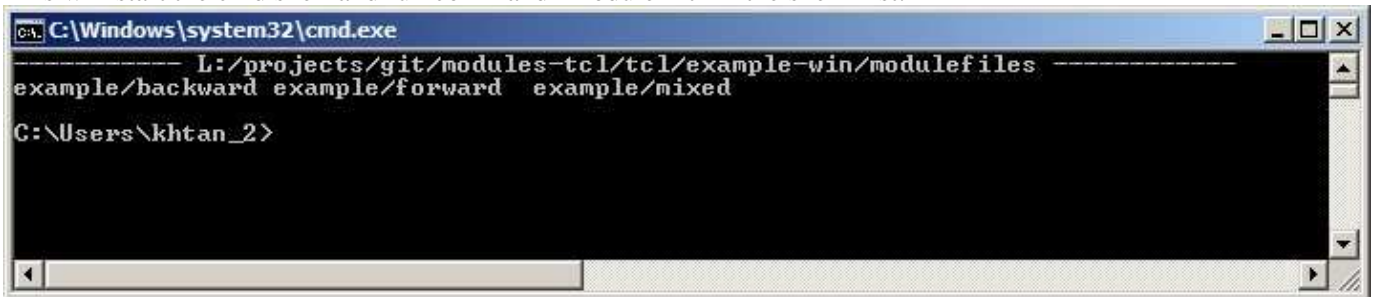


Tip

Once module-tcl is setup correctly, you can also enter the command text "cmd /k moduleinit".



This will start the cmd shell and run command "moduleinit" in the shell first.



There is a reason why you can't use the command "moduleinit" instead of "cmd /k moduleinit". If you tried the former, you'll get an error message that Windows cannot find, for example "L:/projects/git/modules-tcl/tcl/moduleinit.cmd". The Run Dialog does not directly support mixed slashes in the pathname, but cmd does.

This is similar to the case whenever you modify the PATH variable, the cmd shell will see this environment change, but the Run Dialog will not. The Run Dialog only updates its view of PATH once, upon user login.

See [Slashes and spaces in pathname](#) for how module-tcl cmd shell supports forward, backward and mixed slashes in the pathname.

Better moduleinit.cmd

Below is a more realistic moduleinit.cmd. It loads different modules depending on which machine is detected.

```
@echo off
REM -----
REM Script to customize module-tcl for "cmd" shell
REM Load the modulepaths with "module use"
REM -----
```

```

REM -- Check for machine name
if /i %COMPUTERNAME% equ teh goto SETUP-TEH
if /i %COMPUTERNAME% equ kopi-vista goto SETUP-KOPI-VISTA
echo Computer %COMPUTERNAME% not yet supported
goto :EOF

REM -- Setup for machine TEH
:SETUP-TEH
    call module use L:/projects/git/envmodules/modulefiles/testspacesinpath
    call module use L:/projects/git/envmodules/modulefiles/khtan
    goto AVAIL

REM -- Setup for machine KOPI-VISTA
:SETUP-KOPI-VISTA
    call module use H:/projects/git/envmodules/modulefiles/khtan
    goto AVAIL

REM -- Show what modules are available
:AVAIL
    call module avail

```

What's with module-tcl/tcl/init Folder

This is the folder for initializing module-tcl for **Nix* platform. It uses a Makefile and Perl.

The [Windows setup](#) does not use this. However, **Nix* emulators like Cygwin or MKS will most likely make use of this, should they be supported by module-tcl in the future.

Example: Forward slashes in pathnames

The modulefile module-tcl/tcl/example-win/modulefiles/forward uses forward slashes in the pathname of sHello.bat.

Below is a cmd shell transcript showing its use, and the step by step verification.

```

c:\tmp>sHello ❶
sHello
'sHello' is not recognized as an internal or external command,
operable program or batch file.

c:\tmp>module display example/forward ❷
module display example/forward
-----
L:/projects/git/modules-tcl/tcl/example-win/modulefiles/example/forward:

module-what is    TestPathSeparators
append-path      PATH ❸ L:/projects/git/modules-tcl/tcl/example-win/batchfiles/level 1/level
-----

c:\tmp>module load example/forward ❹
module load example/forward

c:\tmp>sHello ❺
sHello
"Spaced Hello from khtan_2 on TEH"

c:\tmp>module unload example/forward ❻
module unload example/forward

c:\tmp>sHello ❼
sHello
'sHello' is not recognized as an internal or external command,

```

operable program or batch file.

- ❶ Confirm sHello.bat is not in PATH
- ❷ Use "module display" to see forward slashed pathname
- ❸ See append-path is forward slashed
- ❹ Load the modulefile
- ❺ sHello.bat is now in PATH and executes
- ❻ Unload the modulefile
- ❼ Confirm sHello.bat is not in PATH

Example: Backward slashes in pathnames

The modulefile module-tcl/tcl/example-win/modulefiles/backward uses backslashes in the pathname of sHello.bat.

Below is a cmd shell transcript showing its use, and the step by step verification.

```
c:\tmp>sHello ❶
sHello
'sHello' is not recognized as an internal or external command,
operable program or batch file.

c:\tmp>module display example/backward ❷
module display example/backward
-----
L:/projects/git/modules-tcl/tcl/example-win/modulefiles/example/backward:

module-what is    TestPathSeparators
append-path      PATH          ❸ L:/projects/git/modules-tcl/tcl\example-win\batchfiles\level 1\level 2\lev

c:\tmp>module load example/backward ❹
module load example/backward

c:\tmp>sHello ❺
sHello
"Spaced Hello from khtan_2 on TEH"

c:\tmp>module unload example/backward ❻
module unload example/backward

c:\tmp>sHello ❼
sHello
'sHello' is not recognized as an internal or external command,
operable program or batch file.
```

- ❶ Confirm sHello.bat is not in PATH
- ❷ Use "module display" to see backslashed pathname
- ❸ See append-path is backslashed
- ❹ Load the modulefile
- ❺ sHello.bat is now in PATH and executes
- ❻ Unload the modulefile
- ❼ Confirm sHello.bat is not in PATH

Example: Mixed slashes in pathnames

The modulefile module-tcl/tcl/example-win/modulefiles/mixed uses mixed slashes in the pathname of sHello.bat.

Below is a cmd shell transcript showing its use, and the step by step verification.

```
c:\tmp>sHello ❶
sHello
'sHello' is not recognized as an internal or external command,
operable program or batch file.

c:\tmp>module display example/mixed ❷
module display example/mixed
-----
L:/projects/git/modules-tcl/tcl/example-win/modulefiles/example/mixed:

module-what is      TestPathSeparators
append-path        PATH ❸ L:/projects/git/modules-tcl/tcl/example-win/batchfiles\level 1/level
-----

c:\tmp>module load example/mixed ❹
module load example/mixed

c:\tmp>sHello ❺
sHello
"Spaced Hello from khtan_2 on TEH"

c:\tmp>module unload example/mixed ❻
module unload example/mixed

c:\tmp>sHello ❼
sHello
'sHello' is not recognized as an internal or external command,
operable program or batch file.
```

- ❶ Confirm sHello.bat is not in PATH
- ❷ Use "module display" to see mixed slashed pathname
- ❸ See append-path is mixed slashed
- ❹ Load the modulefile
- ❺ sHello.bat is now in PATH and executes
- ❻ Unload the modulefile
- ❼ Confirm sHello.bat is not in PATH

How to use module-tcl cmd shell during startup

It is possible for Windows to run your modules cmd file when it starts up.

First, write your module commands into a .cmd or .bat file. Then, put it (or create a shortcut) in the "Program Files/Startup" folder.

When you login to Windows, a cmd shell window with the name of your script will appear and run. This cmd shell will close itself at the end.

Below is an example, named "startup.cmd" to show how to use module-tcl for this purpose.

```
REM -----
REM This is an example of a login script
REM -----
REM --- initialize module
call moduleinit

REM --- load required modulefiles
call module load shells/unxutils
call module load util/graphviz
call module load vc/git
```

```
call module load editors/emacs
```

```
REM --- invoke favourite applications
```

```
call runemacs -g 84x52+0+0
```

```
REM --- open file explorer on favourite locations
```

```
REM      file explorer does not understand forward slashed pathname
```

```
start explorer /e,/root,h:\projects\git\docbookhelp4
```