Remote Shells and Virtual Machines

Version 6.1.1

November 4, 2014

The `"remote-shell"` collection provides tools for running shell commands on a remote or virtual machine, including tools for starting, stopping, and managing VirtualBox virtual-machine instances.

Contents

1 Remote Shells

```
(require remote-shell/ssh)      package: remote-shell-lib
```

```
(remote? v) → boolean?
  v : any/c
```

Returns `#t` if *v* is a remote-host representation produced by `remote`, `#f` otherwise.

```
(remote #:host host
       [#:user user
        #:env env
        #:remote-tunnels remote-tunnels
        #:timeout timeout-secs])         → remote?
  host : string?
  user : string? = ""
  env : (listof (cons/c string? string?)) = '()
                   (listof (cons/c (integer-in 1 65535)
  remote-tunnels :                  (integer-in 1 65535))) = null
  timeout-secs : real? = 600
```

Creates a representation of a remote host. The *host* argument specifies the host for an `ssh` connection. If *user* is empty, then the current user name is used for the remote host.

The *env* argument specifies environment variables to set before running any command on the remote host.

The *remote-tunnels* argument specifies ports to tunnel from the remote host back to the local host. The first port number in each pair is the port number on the remote host, and the second port number is the port that it tunnels to on the local host.

The `timeout` argument specifies a timeout after which a remote command will be considered failed.

```
(ssh remote
     command
    [#:mode mode
     #:failure-log failure-dest
     #:success-log success-dest
     #:show-time? show-time?])
 → (or/c void? boolean? (cons/c boolean? bytes?))
  remote : remote?
  command : (or/c string? path-string?)
  mode : (or/c 'error 'result 'output) = 'error
  failure-dest : (or/c #f path-string?) = #f
  success-dest : (or/c #f path-string?) = #f
  show-time? : any/c = #f
```

Runs a shell command at `remote`, were the `command`s are concatenated (with no additional spaces) to specify the remote shell command. The remote command is implemented with `ssh` as found by `find-system-path`.

If `mode` is `'error`, then the result is `(void)` or an exception is raised if the remote command fails with an connection error, an error exit code, or by timing out. If `mode` is `'result`, then the result is `#t` for success or `#f` for failure. If `mode` is `'cons`, then the result is a pair containing whether the command succeeded and a byte string for the command's output (including error output).

If `failure-dest` is not `#f`, then if the command fails, the remote output (including error output) is recorded to the specified file. If `success-dest` is not `#f`, then if the command fails, the remote output (including error output) is recorded to the specified file.

```
(scp remote source dest [#:mode mode]) → (or/c void? boolean?)
  remote : remote?
  source : path-string?
  dest : path-string?
  mode : (or/c 'error 'result 'output) = 'error
```

Copies a file to/from a remote host. Use `at-remote` to form either the `source` or `dest` argument. The remote command is implemented with `scp` as found by `find-system-path`.

If `mode` is `'error`, then the result is `(void)` or an exception is raised if the remote command fails. If `mode` is `'result`, then the result is `#t` for success or `#f` for failure.

```
(at-remote remote path) → string?
  remote : remote?
  path : path-string?
```

Combines `remote` and `path` to form an argument for `scp` to specify a path at the remote host.

```
(make-sure-remote-is-ready  remote
                           [#:tries tries]) → void?
  remote : remote?
  tries : exact-nonnegative-integer? = 3
```

Runs a simple command at `remote` to check that it receives connections, trying up to `tries` times.

2 Managing VirtualBox Machines

```
(require remote-shell/vbox)      package: remote-shell-lib
```

```
(start-vbox-vm  name
               [#:max-vms max-vms
                #:log-status log-status
                #:pause-seconds pause-seconds
                #:dry-run? dry-run?])         → void?
  name : string?
  max-vms : real? = 1
  log-status : (string? #:rest any/c . -> . any) = printf
  pause-seconds : real? = 3
  dry-run? : any/c = #f
```

Starts a VirtualBox virtual machine *name* that is in a saved, powered off, or running state (where a running machine continues to run).

The start will fail if *max-vms* virtual machines are already currently running. This limit is a precaution against starting too many virtual-machine instances, which can overwhelm the host operating system.

The *log-status* argument is used to report actions and status information.

After the machine is started, `start-vbox-vm` pauses for the amount of time specified by *pause-seconds*, which gives the virtual machine time to find its bearings.

If `dry-run` is `#t`, then the machine is not actually started, but status information is written using *log-status* to report the action that would have been taken.

```
(stop-vbox-vm  name
              [#:save-state? save-state?
               #:log-status log-status
               #:dry-run? dry-run?])      → void?
  name : string?
  save-state? : any/c = #t
  log-status : (string? #:rest any/c . -> . any) = printf
  dry-run? : any/c = #f
```

Stops a VirtualBox virtual machine *name* that is in a running state. If *save-state?* is true, then the machine is put into saved state, otherwise the current machine state is discarded and the machine is powered off.

The *log-status* argument is used to report actions and status information.

If `dry-run` is `#t`, then the machine is not actually started, but status information is written using *log-status* to report the action that would have been taken.

```
(take-vbox-snapshot name snapshot-name) → void?
  name : string?
  snapshot-name : string?
```

Takes a snapshot of a virtual machine (which may be running), creating the snapshot named *snapshot-name*.

```
(restore-vbox-snapshot name snapshot-name) → void?
  name : string?
  snapshot-name : string?
```

Changes the current state of a virtual machine to be the one recorded as *snapshot-name*. The virtual machine must not be running.

```
(delete-vbox-snapshot name snapshot-name) → void?
  name : string?
  snapshot-name : string?
```

Deletes *snapshot-name* for the virtual machine *name*.

```
(exists-vbox-snapshot? name snapshot-name) → boolean?
  name : string?
  snapshot-name : string?
```

Reports whether *snapshot-name* exists for the virtual machine *name*.