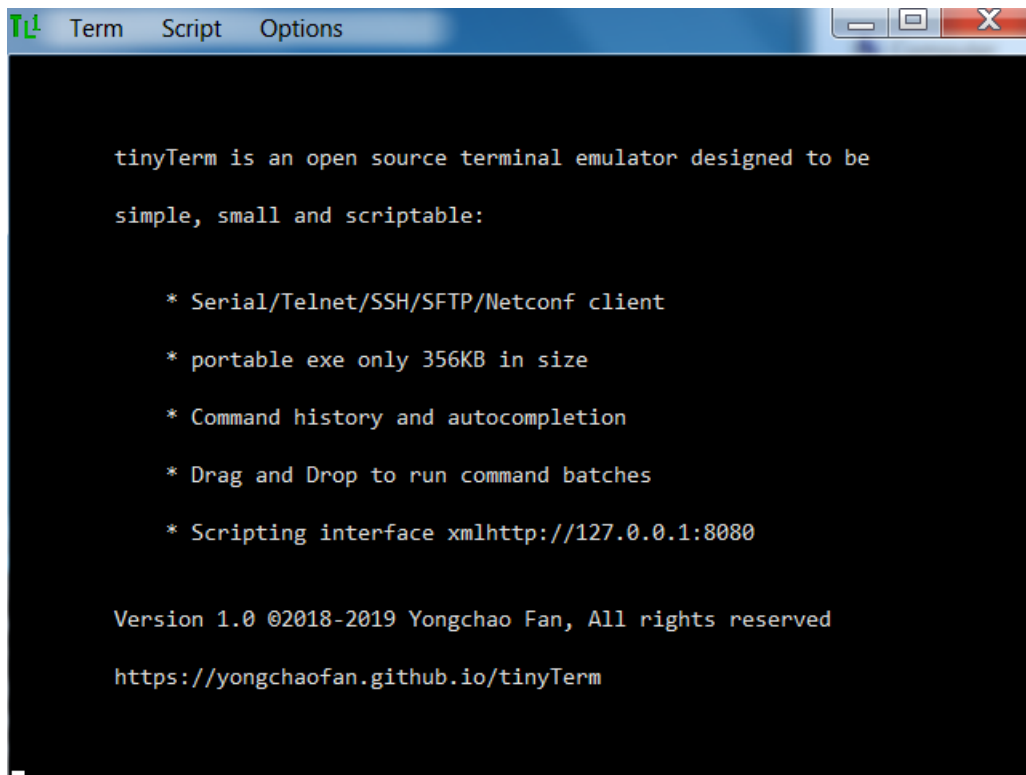


0. Introduction

tinyTerm is a terminal emulator designed to be small, simple and scriptable, and with unique additional features to improve efficiency and effectiveness of command line interface user.

It's designed with the minimalist philosophy, which resulted clean user interface, single executable file 356KB in size, no installation needed, and no extra DLL required



tinyTerm is released under LGPL 3.0 and hosted on github.com

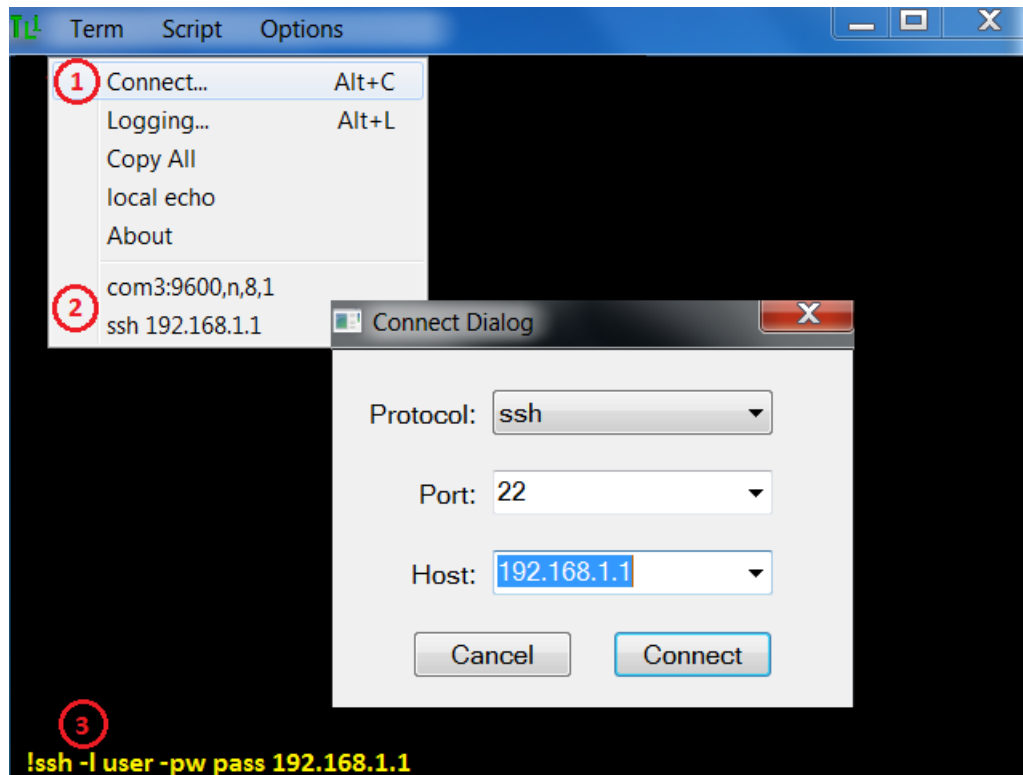
Users go to <https://yongchaofan.github.io/tinyTerm> for downloads and updates

Developers go to <https://github.com/yongchaofan/tinyTerm> for source code, license and issues

1. Making Connection

There are two ways to make a connection in tinyTerm, as shown in the screenshot below.

1. Click "Connect..." from Term menu or Alt+C, the connect dialog will popup, select protocol, select or enter port and destination host in the dialog, then click "Connect"
2. Choose one of the connections used previously from Term menu
3. Type connection command like "!ssh " in autocomplete mode, see next section



Five types of connections supported: serial, telnet, ssh, sftp and netconf.

For serial connections, available serial ports are auto detected and added to the ports drop down list when protocol is changed to serial

For ssh/sftp/netconf connections, command line options are supported

-l username, -pw password, -pp passphrase, -P destination port

Private key based authentication is supported too, key file should be stored in \$USER/.ssh folder, as is the knownhost file for host verification.

2. Sending Command

By default key strokes in the terminal window is sent to remote host directly like any other terminal program.

Autocomplete is a unique feature in tinyTerm, which can be enabled through “Options” menu or Alt-A. Key strokes in autocomplete mode is auto completed using command history, and only sent to host after “Enter” key is pressed. When in Autocomplete mode, cursor changes to a vertical line instead of horizontal bar, and user input are displayed in yellow instead of white. Every command typed in Autocomplete mode is added to command history, and the command history is saved to tinyTerm.hist at exit, then loaded into memory at the next start of tinyTerm.

```
Term Script Options pi@pi3: ~
Adafruit_Python_LED_Backpack flTable Music python27
Desktop fltk-1.3.4 opencv-3.1.0 Templates
dillo-3.0.5 fltk-1.4.x-r13061 ovpn timer
Documents libssh2-1.8.0 Pictures Videos
Downloads libssh2-1.8.1-20180930 Public
pi@pi3:~ $ls_

key strokes in raw mode are sent to host directly

ls -al 68.1.1 key strokes in autocomplete mode
ls -IR :rm.us.to are sent after "Enter" key is pressed

pi@pi3:~ $ls -al
```

When special characters “!,” is typed at the beginning of a command in autocomplete mode, the command will be executed by tinyTerm instead of sending to remote host, for functions like making connection, search scroll buffer, set options, scp file transfer or ssh tunneling etc. See Appendix A for list of supported special commands.

Note1: since the command history file tinyTerm.hist is just a plain text file, user can edit the file outside of tinyTerm to put additional commands in the list for command auto-completion. For example put all the TL1 commands that are used often in the history list to use as a dictionary.

Note 2: connection command started with “!” in the history file will be added to “Term” menu at the start of tinyTerm, so that connections can be made easily through menu clicks

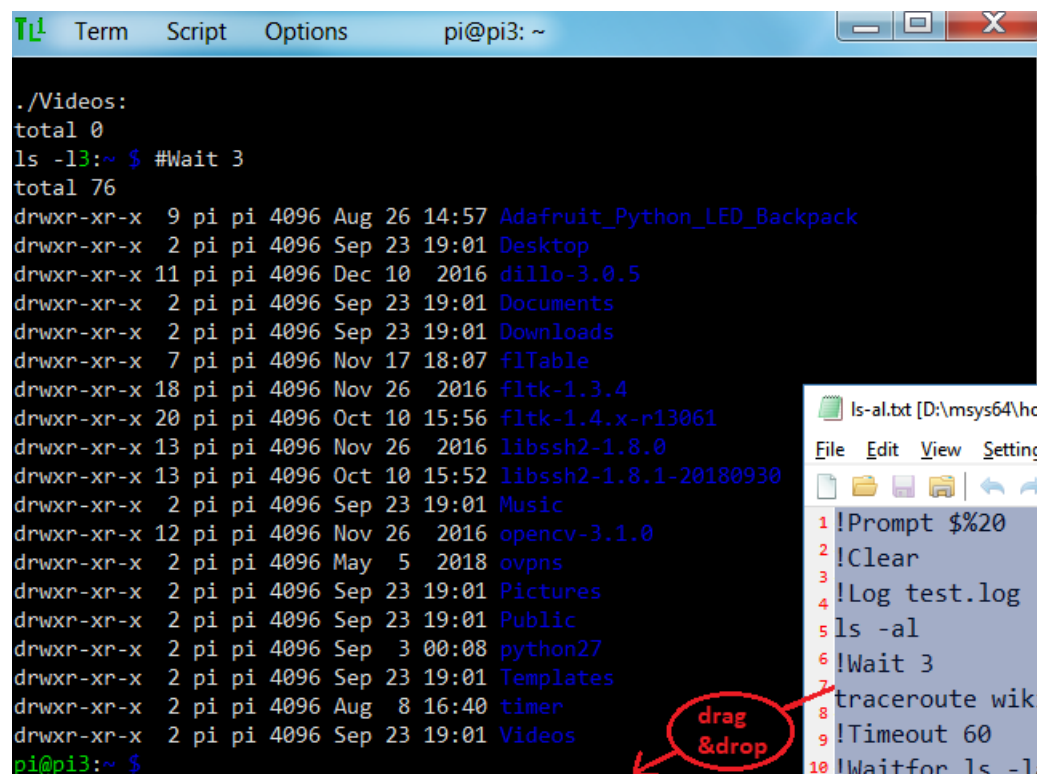
3. Running Script

Scripting is the way of choice to automate repetitive tasks, tinyTerm is designed to make scripting as easy as possible.

The simplest type of script is just a list of commands in plain text format, either drag&drop a block of text to the cursor line in autocomplete mode, or drag&drop a text file to the terminal window, or "Run..." a text files from the "Script" menu, tinyTerm will send commands in the list one by one.

Note that contrary to popular terminal programs, tinyTerm will not send all commands at once, risking overflow of the input buffer at remote host or network device, instead tinyTerm will wait for the prompt string from remote host before sending each command. Most command line interface system uses a prompt string to tell user it's ready for the next command, for example ">" or "\$" used by Cisco routers.

tinyTerm will auto detect the prompt string used by remote host when user is typing commands interactively, and use the detected prompt string during scripting. Additionally, prompt string can be set in the script using special command "!Prompt {str}", refer to appendix A for details and other special commands supported for scripting.



The screenshot shows the tinyTerm application window with a terminal session on a Raspberry Pi. The terminal displays the output of a directory listing command, showing various files and folders. A red circle with the text "drag & drop" points to the script editor window, which is open and shows a list of commands to be executed. The script editor window has a menu bar with "File", "Edit", "View", and "Setting". The script content is as follows:

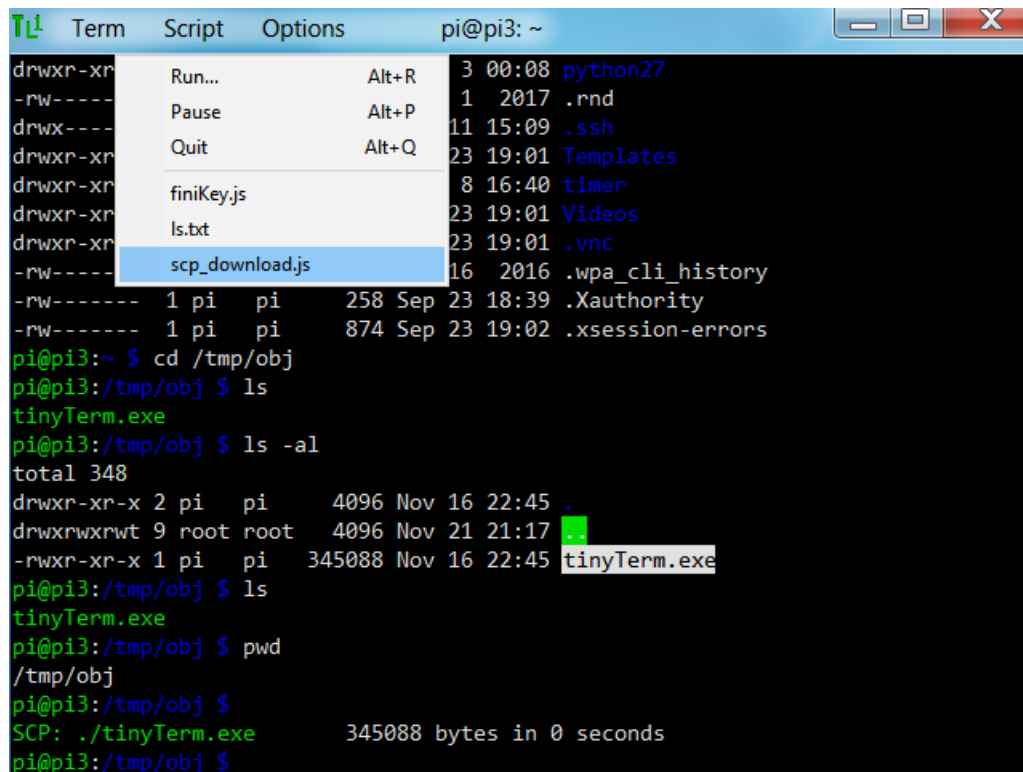
```
1 !Prompt $%20
2 !Clear
3
4 !Log test.log
5 ls -al
6 !Wait 3
7 traceroute wik
8
9 !Timeout 60
10 !Waitfor ls -l
```

4. Advanced Scripting

tinyTerm has a xmlhttp interface built in at 127.0.0.1:8080 for advanced scripting. VBScript and JavaScript are two of the scripting languages that can take advantage of the xmlhttp interface. Files in the script folder will be listed in the script menu, select one to execute, or use "Run..." to choose a script file from the file system.

The screen capture below shows the execution of scp_download.js from script menu, which retrieves the selected filename "tinyTerm.exe" and uses "!scp" command to download from remote host, all operation are performed using xmlhttp://127.0.0.1:8080 to send command through tinyTerm. When multiple instance of tinyTerm are started, each instance will use a unique port number starting from 8080 and counting up, 8081, 8082...etc. Special commands supported on the xmlhttp interface:

!Disp {str}	Display {str} in terminal window
!Recv	Return scroll buffer content since last Disp/Recv/Send command
!Send {cmd}	Send command and return immediately
!Selection	Get current text selection



The screenshot shows a terminal window titled "Term Script Options pi@pi3: ~". A menu is open with options: Run... (Alt+R), Pause (Alt+P), Quit (Alt+Q), finiKey.js, ls.txt, and scp_download.js (highlighted). The terminal output shows the user navigating to /tmp/obj, listing files (including tinyTerm.exe), and then executing the scp_download.js script. The script successfully downloads tinyTerm.exe from a remote host, showing "SCP: ./tinyTerm.exe 345088 bytes in 0 seconds".

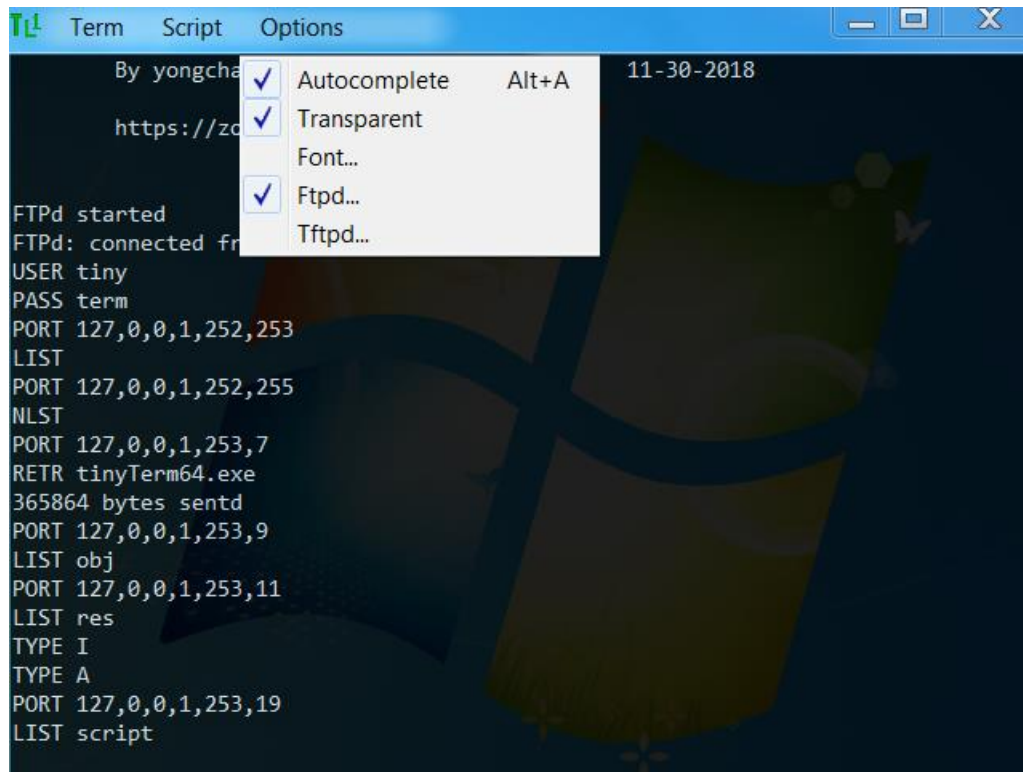
```
pi@pi3:~ $ cd /tmp/obj
pi@pi3:/tmp/obj $ ls
tinyTerm.exe
pi@pi3:/tmp/obj $ ls -al
total 348
drwxr-xr-x 2 pi pi 4096 Nov 16 22:45 .
drwxrwxrwt 9 root root 4096 Nov 21 21:17 ..
-rwxr-xr-x 1 pi pi 345088 Nov 16 22:45 tinyTerm.exe
pi@pi3:/tmp/obj $ ls
tinyTerm.exe
pi@pi3:/tmp/obj $ pwd
/tmp/obj
pi@pi3:/tmp/obj $
SCP: ./tinyTerm.exe 345088 bytes in 0 seconds
pi@pi3:/tmp/obj $
```

5. Options

While the user interface is minimal, there are a few options to customize through Options menu, like font face and font size, window transparency for example. All of the options can be set with command through the editor line or scripts. See appendix A for list of all supported commands.

A built in FTP server can be used for simple file transfer tasks, like software download to network devices. Only one user name "tiny" is allowed to login, with password "term". For security, user session to the FTP server is timed out in 1 minute without action, and FTP server will time out in 15 minutes without active connection.

Similarly a built in TFTP server can be used for file transfer with simpler devices like cable modems. TFTP server times out after 5 minutes.



Appendix A. List of special command for editor line and scripting

Connection

!com3:9600,n,8,1	serial connection to port com3 using settings 9600,n,8,1
!telnet 192.168.1.1	telnet to 192.168.1.1
!ssh 192.168.1.1	ssh to host 192.168.1.1
!sftp -l admin 192.168.1.1	sftp to host 192.168.1.1 with user admin
!netconf -P 830 192.168.1.1	netconf to port 830 of host 192.168.1.1
!disconn	disconnect from current connection
!{DOS command}	execute command and display result, e.g. ping 192.168.1.1

Options

!TermSize 100x40	set terminal window size to 100 columns x 40 rows
!Transparency 192	set window transparency level to 192/255
!FontFace Consolas	set font face to "Consolas"
!FontSize 18	set font size to 18

Scripting

!Clear	set clear scroll back buffer
!Prompt \$%20	set command prompt to "\$ ", for batch command execution
!Timeout 30	set time out to 30 seconds for batch command execution
!Wait 10	wait 10 seconds during batch command execution
!Waitfor 100%	wait for "100%" from during batch command execution
!Loop 2	repeat two times from start of script
!Log test.log	start/stop logging with log file test.log

Advanced Scripting

!Disp test case !1	display "test case !1" in terminal window
!Send ping 192.168.1.1	send "ping 192.168.1.1" to host
!Recv	get all text received since last Disp/Send/Recv command
!Echo	toggle local ech on/off
!Selection	get current selected text

!Ftpd c:/tmp	start/stop ftp server using c:/tmp as root directory
!Tftpd c:/tmp	start/stop tftp server using c:/tmp as root directory

!scp test.txt :test1.txt	secure copy local file test.txt to remote host as test1.txt
!scp :test1.txt d:/	secure copy remote file test1.txt to local d:/test1.txt

!tun	list all ssh2 tunnels
!tun 3256	terminal ssh2 tunnel number 3256
!tun 127.0.0.1:2222 127.0.0.1:22	start ssh2 tunnel from localhost port 2222 to remote host port 22