UseCase_4_Command_Line_Interface

TeamSTARS "tsWxGTUI_PyVx" Toolkit
with **Python**™ 2x & **Python**™ 3x based
Command Line Interface (CLI)
and **"Curses"**-based **"wxPython"**-style
Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-color (xterm-family) and non-color (vt100-family) terminals and terminal emulators.
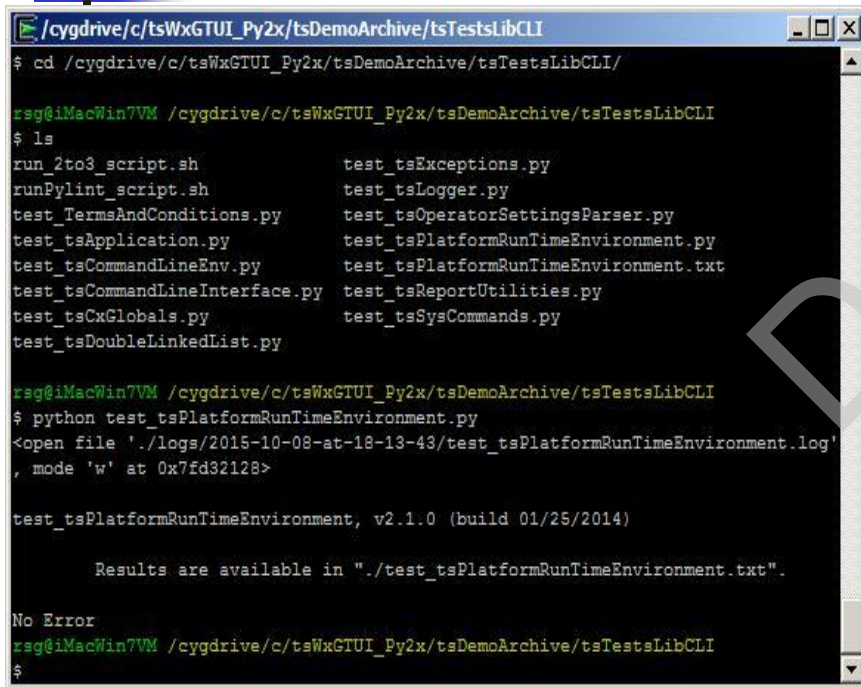
Table of Contents *(with slide show* <u>Hyperlinks</u>*)*

- <u>tsLibCLI</u> --- Library of building block components

- <u>tsToolsCLI</u> --- Collection of development productivity tools

- <u>Distributed System Procedures</u> --- Commands which operate one or more remote computer systems from the terminal attached to the local computer system. Each operating system's on-line manual pages provide applicable details.

Command Line Interface (CLI) Overview

- **Output to the User:**
  - A chronological sequence of lines of text written from top to bottom and then scrolling off the top as each new line is written to the bottom of the terminal display.
- **Input from the User:**
  - Via a computer terminal keyboard with input echoed to the display below the previous output.
    - The **cd** command, also known as **chdir**, changes the directory as specified.
    - The **ls** command lists files in the directory.
    - The **python** command executes the named program which displays the location of its results before terminating.

CLI Building Blocks (tsLibCLI *full listing*)

- **Application Building Blocks**
  - tsCommandLineInterface
  - tsDoubleLinkedList
  - tsOperatorSettingsParser
  - tsReportUtilities
- **Application Diagnostics**
  - tsExceptions
  - tsLogger

- **Application Configuration**
  - tsCxGlobals
  - tsGistGetTerminalSize
  - tsPlatformRunTimeEnviroment
- **Application Launchers**
  - tsApplication
  - tsCommandLineEnv
  - tsSysCommands

tsLibCLI
Organization and Usage

- This library of building blocks is organized, by the functional scope of each component, into a collection of Python "modules".

- When appropriate, tsLibCLI modules may import and use the services of:
  - any other tsLibCLI module
  - any module listed in the Python Global Module Index

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

tsApplication.py Module

- Module enables the application program launched by an operator via a Command Line Interface (CLI) to also initialize, configure and use the same character-mode terminal with a Graphical-style User Interface (GUI).
    - Module registers and validates:
        - operator application settings inputs from command line keyword-value pairs and positional arguments.
        - instantiation settings input from applications via caller parameter list.
- Module is the base class, for **tsCommandLineEnv** which itself is the base class for **tsWxMultiFrameEnv**.

tsCommandLineEnv.py Module

- Class to initialize and configure the application program launched by an operator.
  - Class delivers those keyword-value pair options and positional arguments specified by the application, in its invocation parameter list.
  - Class wraps the Command Line Interface application with exception handlers to control exit codes and messages that may be used to co-ordinate other application programs.

tsCommandLineInterface.py Module

- Class establishes methods that:
  - prompt or re-prompt the operator for input
  - validate that the operator has supplied the expected number of inputs and that each is of the expected type.\
  - does NOT validate the input value

tsCxGlobals.py Module

- Module to establish configuration constants and macro-type functions for the Command Line Interface mode of the "tsWxGTUI" Toolkit.

  - Module provides a theme-based mechanism for modifying/restoring configuration constants as appropriate for various user roles and activities.

tsDoubleLinkedList.py Module (Table of Contents)

- Class to establish a general purpose representation of a linked list with forward and backward pointers
  - Class provides methods to append, insert, delete and access the ordered list of entries.

tsException.py Module

- **Class to define and handle error exception events.**
  - Class maps run time exception types into 8-bit exit codes
  - Class prints associated exception diagnostic message and trace back information

tsGistGetTerminalSize.py Module

- **Third-Party Module, derived from "terminalsize.py" by Justin T. Riley:**
  - Module acquires the character size of the Python console window as a Python tuple (width, height) on host operating systems (such as Linux, Mac OS X, Microsoft Windows and Unix).

tsLogger.py Module

- Class emulates a subset of Python logging API.

  - Class defines and handles prioritized, time and date stamped event message formatting and output to files and devices:

    - Log files are organized in a date and time stamped directory named for the launched application.

    - Log devices include the Unix-type **syslog**, **stderr**, **stdout** and **stdscr** (the Curses display screen).

- Class also supports "wxPython"-style logging of assert and check case results.

tsOperatorSettingsParser.py Module

- Class to parse the command line entered by the operator of an application program:
  - Platform-independent parsing algorithm extracts and returns the Keyword-Value pair Options and Positional Arguments that will configure and control the application during its execution.
  - Platform–specific parsing algorithm applies the standard Python library module ("**argparse**", "**optparse**" or "**getopt**") appropriate for the active Python version.

tsPlatformRunTimeEnvironment.py Module

- Class to capture current hardware, software and network information about the run time environment for the user process.
  - Host processor hardware support includes various releases of Arm, x86, PowerPC, SPARC and others.
  - Host operating system software support includes various releases of Cygwin, Linux, Mac OS X, Unix, Windows and others.
  - Host virtual machine software support includes various releases of Java and Python.
  - Network identification support includes host name, aliases and ip-address list.
  - Environment Variable support includes user, session, shell, path and time zone
- It makes this information available via a file (default is "./PlatformRunTimeEnvironment.txt").

tsReportUtility.py Module

- Class defining methods used to format information for the operator's display and log files:
  - Convert file size from numeric and string format with optional kilo-, mega-, giga-, tera-, peta-, exa-, zeta- and yotta-byte units
  - Convert time between string and seconds formats.
  - Construct a formatted log of multi-level nested Python dictionary contents
  - Construct a string of title and white space to separate one section of text from another.
  - Construct a string of white space appropriate for indenting level.
  - Construct the path to the next log file.
  - Create test summary after elapsed time with statistics details on the
    - Number of (total test runs, passing test run subtotal and failing test run subtotal)
    - Timestamp (startup, shutdown and elapsed)

tsSysCommands.py Module

- ## Class definition and methods for:
  - Issuing shell commands to the host operating system
  - Receiving responses from the host operating system
  - Wrapping and using appropriate Python sub-process module methods.

Software Development Productivity Tools (tsToolsCLI *full listing*)

- **Developer Tools**
  - tsStripComments
  - tsStripLineNumbers
  - tsTreeCopy
  - tsTreeTrimLines.py
- **Troubleshooter Tools**
  - tsPlatformQuery

- **Project Tools**
  - tsLinesOfCodeProjectMetrics

tsStripComments.py Module

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.
    - The application transforms an annotated, development version of a directory containing subdirectories and Python source files into an unannotated copy.
    - The unannotated copy is intended to conserve storage space when installed in an embedded system.
    - The transformation involves stripping comments and "docstrings" by detokenizing a tokenized version of each Python source file.
    - Non-Python files are trimmed of trailing whitespace.

tsStripLineNumbers.py Module

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.

  - Application strips line numbers from source code (such as from annotated documentation listings) that would not be required as reference points for conditional branching.

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

tsTreeCopy.py Module (Table of Contents)

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.

  - Application copies the files and directories contained in an input source directory to an output target directory.

tsTreeTrimLines.py Module ([Table of Contents])

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.

  - Application copies the files and directories contained in an input source directory to an output target directory after stripping superfluous white space (blanks) from end of each line.

tsPlatformQuery.py Module

- Python application program, launched via a Command Line Interface (CLI) with options for the operator to designate input and output directories.

  - Application uses tsPlatformRunTimeEnvironment module to capture and report current hardware and software information about the run time environment available to computer programs.

tsLinesOfCodeProjectMetrics.py Module and Building Block Modules (Table of Contents)

- Python application program and building block modules that:
  - Launches via Command Line Interface (CLI) with options designating input directory and output file
  - Scans an operator designated file directory tree containing the source files, in one or more programming language specific formats (such as Ada, Assembler, C/C++, Cobol, Fortran, PL/M, Python, Text, and various command line shells).
  - Accumulates and reports the number of code lines, blank/comment lines, words and characters for individual files, language subtotals and project totals.
  - Generates reports of software project progress and metrics for the software development project (such as labor, cost or contributed value, schedule and lines of code per day productivity).

Distributed System Procedures

- **Monitoring and Control Procedure**
  - Uses the Command Line & Graphical User Interfaces to operate the mission critical equipment for the acquisition of sensor and operator input data and for the output of control signals, setpoints, constraints and operator status updates via the embedded computer system.

- **File Transfer Procedure**
  - Uses the Command Line Interface to captures the most recent Monitoring and Control operation data and transcript logs, before it can be overwritten and/or mixed with new data, in preparation for off-line troubleshooting and analysis of operation and performance.
  - Updates computer software configuration data and source code in preparation for the next Monitoring and Control operation.

Monitoring and Control Procedure

- Once you've logged into your local computer, you may then use one or more secure shells ("**ssh**") or non-secure shells ("**rsh**") provided by the local operating system to:
  - Monitor and control the local computer from the convenience of your local character-mode computer terminal display, keyboard and mouse.
  - Optionally monitor and control one or more remote computers also from the convenience of your local character-mode computer terminal display, keyboard and mouse, with greater speed and efficiency than possible with the larger communication traffic associated with pixel-mode.
  - Use the local computer's character-mode command line interface shell to:
    - Launch one or more of the local computer's:
      - Shell scripts
      - Tools
      - Application programs (including any of the TeamSTARS "tsWxGTUI_PyVx" Toolkit's Python based Command Line Interface or curses-based Graphical User Interface ones).
    - Login to one or more remote computers and then use the remote computer's character-mode command line interface shell to:
      - Launch one or more of the remote computer's:
        - Shell scripts
        - Tools
        - Application programs (including any of the TeamSTARS "tsWxGTUI_PyVx" Toolkit's Python based Command Line Interface or curses-based Graphical User Interface ones).

Secure (Remote) Shell (SSH)

SSH(1)   BSD General Commands Manual   SSH(1)   DESCRIPTION

NAME
    ssh   OpenSSH SSH client (remote login program)

SYNOPSIS
    ssh [-1246AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
       [-D [bind_address:]port] [-E log_file] [-e escape_char]
       [-F configfile] [-I pkcs11] [-i identity_file] [-L address]
       [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port]
       [-Q cipher | cipher-auth | mac | kex | key | protocol-version]
       [-R address] [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
       [user@]hostname [command]

ssh (SSH client) is a program for logging into a remote machine and for executing commands on a remote machine. It is intended to replace rlogin and rsh, and provide secure encrypted communications between two untrusted hosts over an insecure network. X11 connections, arbitrary TCP ports and UNIX-domain sockets can also be forwarded over the secure chanânel.

ssh connects and logs into the specified hostname (with optional user name). The user must prove his/her identity to the remote machine using one of several methods depending on the protocol version used (see below).

If command is specified, it is executed on the remote host instead of a login shell.

File Transfers (Table of Contents)

- Once you've logged into your local computer, you may then use one or more secure file transfer utilities ("**sftp**") or non-secure file transfer utilities ("**ftp**") provided by the local and remote operating systems to:
  - Prepare remote computer platform for Monitoring and Control operation by copying source code files and directories from the local to the remote computer.
  - Prepare for analysis of Monitoring and Control operation by copying data and log files and directories from the remote to the local computer.

Secure File Transfer Protocol (SFTP)

SFTP(1)   BSD General Commands Manual   SFTP(1)

NAME

    sftp   secure file transfer program

SYNOPSIS

    sftp [-1246aCfpqrv] [-B buffer_size] [-b batchfile] [-c cipher]
      [-D sftp_server_path] [-F ssh_config] [-i identity_file] [-l limit]
      [-o ssh_option] [-P port] [-R num_requests] [-S program]
      [-s subsystem | sftp_server] host
    sftp [user@]host[:file ...]
    sftp [user@]host[:dir[/]]
    sftp -b batchfile [user@]host

DESCRIPTION

    sftp is an interactive file transfer program, similar to ftp(1), which
    performs all operations over an encrypted ssh(1) transport. It may also
    use many features of ssh, such as public key authentication and compres-
    sion. sftp connects and logs into the specified host, then enters an
    interactive command mode.

The second usage format will retrieve files automatically if a non-inter-
active authentication method is used; otherwise it will do so after suc-
cessful interactive authentication.

The third usage format allows sftp to start in a remote directory.

The final usage format allows for automated sessions using the -b option.
In such cases, it is necessary to configure non-interactive authentica-
tion to obviate the need to enter a password at connection time (see
sshd(8) and ssh-keygen(1) for details).

Since some usage formats use colon characters to delimit host names from
path names, IPv6 addresses must be enclosed in square brackets to avoid
ambiguity.

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon

Non-Secure Remote Shell (RSH) (Table of Contents)

RSH(1)   BSD General Commands Manual   RSH(1)

NAME

   rsh   remote shell

SYNOPSIS

   rsh [-Kdnx] [-l username] host [command]

DESCRIPTION

   Rsh executes command on host.

   Rsh copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command; rsh normally terminates when the remote command does. The options are as follows:

   -d   The -d option turns on socket debugging (using setsockopt(2)) on the TCP sockets used for communication with the remote host.

   -l   By default, the remote username is the same as the local username. The -l option allows the remote name to be specified.

   -n   The -n option redirects input from the special device /dev/null (see the BUGS section of this manual page).

If no command is specified, you will be logged in on the remote host using rlogin(1).

Shell metacharacters which are not quoted are interpreted on local machine, while quoted metacharacters are interpreted on the remote machine. For example, the command

   rsh otherhost cat remotefile >> localfile

appends the remote file remotefile to the local file localfile, while

   rsh otherhost cat remotefile ">>" other_remotefile

appends remotefile to other_remotefile.

Non-Secure File Transfer Protocol (FTP)

FTP(1)   User Commands   FTP(1)

NAME
    ftp   File Transfer Protocol client.

SYNOPSIS
    ftp [OPTION...] [HOST [PORT]]

DESCRIPTION
    Remote file transfer.

    -d, --debug
        set the SO_DEBUG option

    -g, --no-glob
        turn off file name globbing

    -i, --no-prompt
        do not prompt during multiple file transfers

-n, --no-login
    do not automatically login to the remote system

-p, --prompt[=PROMPT]
    print a command line PROMPT (optionally), even if not on a tty

-t, --trace
    enable packet tracing

-v, --verbose
    verbose output

-?, --help
    give this help list

--usage
    give a short usage message

-V, --version
    print program version

Draft

TeamSTARS "tsWxGTUI_PyVx" Toolkit
prepared & presented by Richard S. Gordon