Acceptance criteria for the lib/runcmd Subplot library

The Subplot project

2021-10-21 11:32

Contents

1 Introduction

The Subplot[1] library `runcmd` for Python provides scenario steps and their implementations for running Unix commands and examining the results. The library consists of a bindings file `lib/runcmd.yaml` and implementations in Python in `lib/runcmd.py`. There is no Bash version.

This document explains the acceptance criteria for the library and how they're verified. It uses the steps and functions from the `lib/runcmd` library. The scenarios all have the same structure: run a command, then examine the exit code, standard output (stdout for short), or standard error output (stderr) of the command.

The scenarios use the Unix commands `true` and `false` to generate exit codes, and `echo` to produce stdout. To generate stderr, they use the little helper script below.

File: **err.sh**

```
#!/bin/sh
echo "$@" 1>&2
```

2 Check exit code

These scenarios verify the exit code. To make it easier to write scenarios in language that flows more naturally, there are a couple of variations.

2.1 Successful execution

when I run **true**
then exit code is **0**
and command is successful

2.2 Successful execution in a sub-directory

given a directory **xyzzy**
when I run, in **xyzzy**, **pwd**
then exit code is **0**
then command is successful
then stdout contains "**/xyzzy**"

2.3 Failed execution

when I try to run **false**
then exit code is not **0**
and command fails

[1] `https://subplot.liw.fi/`

2.4 Failed execution in a sub-directory

given a directory **xyzzy**
when I try to run, in **xyzzy**, **false**
then exit code is not **0**
and command fails

3 Check we can prepend to $PATH

This scenario verifies that we can add a directory to the beginning of the PATH environment variable, so that we can have `runcmd` invoke a binary from our build tree rather than from system directories. This is especially useful for testing new versions of software that's already installed on the system.

given executable script **ls** from **ls.sh**
when I prepend **.** to PATH
when I run **ls**
then command is successful
then stdout contains "**custom ls, not system ls**"

File: **ls.sh**

```
#!/bin/sh
echo "custom ls, not system ls"
```

4 Check output has what we want

These scenarios verify that stdout or stderr do have something we want to have.

4.1 Check stdout is exactly as wanted

Note that the string is surrounded by double quotes to make it clear to the reader what's inside. Also, C-style string escapes are understood.

when I run **echo hello, world**
then stdout is exactly "**hello, world\n**"

4.2 Check stderr is exactly as wanted

given helper script **err.sh** for runcmd
when I run **sh err.sh hello, world**
then stderr is exactly "**hello, world\n**"

4.3 Check stdout using sub-string search

Exact string comparisons are not always enough, so we can verify a sub-string is in output.

when I run **echo hello, world**
then stdout contains "**world\n**"
and exit code is **0**

4.4 Check stderr using sub-string search

given helper script **err.sh** for runcmd
when I run **sh err.sh hello, world**
then stderr contains "**world\n**"

4.5 Check stdout using regular expressions

Fixed strings are not always enough, so we can verify output matches a regular expression. Note that the regular expression is not delimited and does not get any C-style string escaped decoded.

when I run **echo hello, world**
then stdout matches regex **world$**

4.6 Check stderr using regular expressions

given helper script **err.sh** for runcmd
when I run **sh err.sh hello, world**
then stderr matches regex **world$**

5 Check output doesn't have what we want to avoid

These scenarios verify that the stdout or stderr do not have something we want to avoid.

5.1 Check stdout is not exactly something

when I run **echo hi**
then stdout isn't exactly "**hello, world\n**"

5.2 Check stderr is not exactly something

given helper script **err.sh** for runcmd
when I run **sh err.sh hi**
then stderr isn't exactly "**hello, world\n**"

5.3 Check stdout doesn't contain sub-string

when I run **echo hi**
then stdout doesn't contain "**world**"

5.4 Check stderr doesn't contain sub-string

given helper script **err.sh** for runcmd
when I run **sh err.sh hi**
then stderr doesn't contain "**world**"

5.5 Check stdout doesn't match regular expression

when I run **echo hi**
then stdout doesn't match regex **world$**

5.6 Check stderr doesn't match regular expressions

given helper script **err.sh** for runcmd
when I run **sh err.sh hi**
then stderr doesn't match regex **world$**