|  |
| --- |
| System Cleanup |

Let’s say that you have many left-running, a few ßash-shells – iTerm2 instances or macOS Terminal instances.

They are there, even if the macOS GUI Dock option named Show all Windows claims otherwise (and you’ve closed the Terminal Window, but the Terminal’s Process lingers).

A screenshot of a cell phone

Description automatically generated

Above, the macOS Activity Monitor reports the PIDs for the stray/errant process(es).

Below is the same information; however, this info can be redirected to a flat file for further examination, and for possible eradication of all/part of the set of PIDs.

A screenshot of a computer

Description automatically generated

Or, maybe you’ve abandoned a few daemons, or left open a few tail commands to logs, tail sessions which you can terminate, but leave the log file alone (of course).

A screenshot of a cell phone

Description automatically generated



Here is a safe (lists-only) ßash function that finds the PIDs for all running instances of a utility/application by looking for its logical, runtime image name.

You can convert it into a ßash script as well (remove the **bold text**).

Reserved words are blue.

**function pids4image() {**

export token=$1

export incominglen=$(echo "$1" | awk '{print length}')

split=$((incominglen-1))

first=${token:0:split}

last=${token:split:1}

composite=$first[$last]

ps auxx | grep $composite | grep -v grep

if [ -f "$1" ];then

rm $1

fi

ps auxx | grep "$1" | grep -v grep > "$1"

while IFS=' ' read -r line || [[ -n "$line" ]]; do

export PID=2

PID="$(echo $line | cut -d " " -f $PID)"

echo "${PID}"

done < "$1"

if [ -f "$1" ];then

rm $1

fi

**}}}}**

****

**A screenshot of a cell phone

Description automatically generated**