72 lines (62 sloc) | 2.46 KB

```
'use strict';
var execSync = require('child_process').execSync;
var mainPid = require('yargs').argv._[0];

class Process {
  constructor(pid, name, level=0) {
    [this.pid, this.name, this.level, this.children] = [pid, name, level, []];
  }
}

let PsTree = {

  // setup children properties of the given processObj;
  _buildProcessTree(processObj) {
    (typeof processObj === 'number') && (processObj = this.getProcessTree(processObj));
    try {
      let lines = execSync(`pgrep -lfP ${processObj.pid}`).toString().split('\n').filter(e => e);
      lines.forEach(line => {
        let [_, pid, name] = line.match(/^(\d+) (.*)$/);
        processObj.children.push(new Process(pid, name, processObj.level + 1));
        processObj.children.forEach(el => this._buildProcessTree(el));
      });
    } catch(e) {} // pgrep fails when no children found
    return processObj;
  },

  _txtProcessTree(processObj) {
    (typeof processObj === 'number') && (processObj = this.getProcessTree(processObj));
    let prefix = Array(processObj.level).fill('  ').join('');
    let name = processObj.name.length > 80 ? processObj.name.substring(0, 77)+' ...': processObj.name;
    let output = `${prefix} * ${processObj.pid} ${name}\n`;
    processObj.children.forEach(child => output += this._txtProcessTree(child));
    return output;
  },

  getPids(processObj, pidsByLevel = []) {
    (typeof processObj === 'number') && (processObj = this.getProcessTree(processObj));
    pidsByLevel[processObj.level] = pidsByLevel[processObj.level] || [];
    pidsByLevel[processObj.level].push(processObj.pid);
    processObj.children.forEach(child => this.getPids(child, pidsByLevel));
    return pidsByLevel;
  },

  getProcessTree(pid) {
    try {
      let psOutput = execSync(`ps -p ${pid} -o "pid=,command="`).toString().trim();
      let [_, processId, processName] = psOutput.match(/^(\d+) (.*)$/);
      let processObj = new Process(processId, processName);
      this._buildProcessTree(processObj);
      return processObj;
    } catch(e) {
      console.error('Invalid process id.', e.message);
      process.exit(2);
    }
  },

  treeKill(pid) {
    let pids = this.getPids(pid).reduce( (s, e) => s.concat(e)).reverse();
    execSync(`kill -9 ${pids.join(' ')}`);
    return `killed ${pids.join(' ')}`;
  },

  show(pid) {
    console.log( this._txtProcessTree(pid) );
  }
}

if (require.main === module) {
  mainPid ? PsTree.show(mainPid) : console.error('invalid process id') && process.exit(1);
}

module.exports = PsTree;
```