Python Programming III
Classes and Objects, Simple Graphics and GUI
Lab

Objectives

- Define a Python Class
- Create Python Object
- Use class functions
- Define class methods
- Explain inheritance

Table of Contents

1.

Background Reading

Textbook:
How to Think Like a Computer Scientist
Download under the GNU Free Documentation License using following link:
www.greenteapress.com/thinkpython/thinkCSpy.pdf

Read chapters 12,13,14

Notes common to all lab and home assignment problems

For every lab and home assignment, all work should go into your personal repository, subdirectory named mXX, where XX stands for the module number. For each problem, carefully name the program as described. The programs are extracted from your repository by a Python script, and errors in the program name will result in the instructor never seeing your program, and your mark for it will be ZERO!

There are always many ways how to solve a programming problem, and usually one or two ways which are fast, compact and elegant.

Make sure to push your work to the server often, and have pushed the working version of the program by the deadline specified. The script extracting your programs from your repository will be run at any time after the deadline.

Lab specific intro

In this lab, we will read and parse contents of the linux /proc pseudo-file system, namely the per-process part, /proc/[pid]/stat. Further, we will utilize the python graphics library PIL and generate a picture. Then we will design simple graphical user interface to interact with the application, using GTK3 and glade GUI designer.

Problem 1

Write Python program **`m04p01.py`** which will create class LinuxProcess. Write methods to extract the following fields, apply the methods and print the results following this example:

```
          name:           m04p01.py
       rss_lim: 0xffffffffffffffffL
    start_code:             0x400000
      end_code:             0x6bb0f4
   start_stack:       0x7fffdd658190
           esp:       0x7fffdd657a18
           eip:       0x7f3468dad810
    start_data:             0x8bbdc0
      end_data:             0x9303f4
```

```
start_brk:          0x2872000
arg_start:     0x7fffdd658661
  arg_end:     0x7fffdd65867d
env_start:     0x7fffdd65867d
  env_end:     0x7fffdd658fec
```

Problem 2

Write Python program **`m04p02.py`** which will create class LinuxProcList. This class will read all the running processes from /proc/[pid]/stat and generate internal structure which will represent the process tree. It will also read the command lines used to invoke each process. You will need to read at least the files /proc/[pid]/stat and /proc/[pid]/cmdline. The methods in the class should be at least:

`LinuxProcList.proclist()`

returns list of all process ids

`LinuxProcList.cmdline(pid)`

returns string containing the command line for given process, or None

`LinuxProcList.children(pid)`

returns list of children of given process

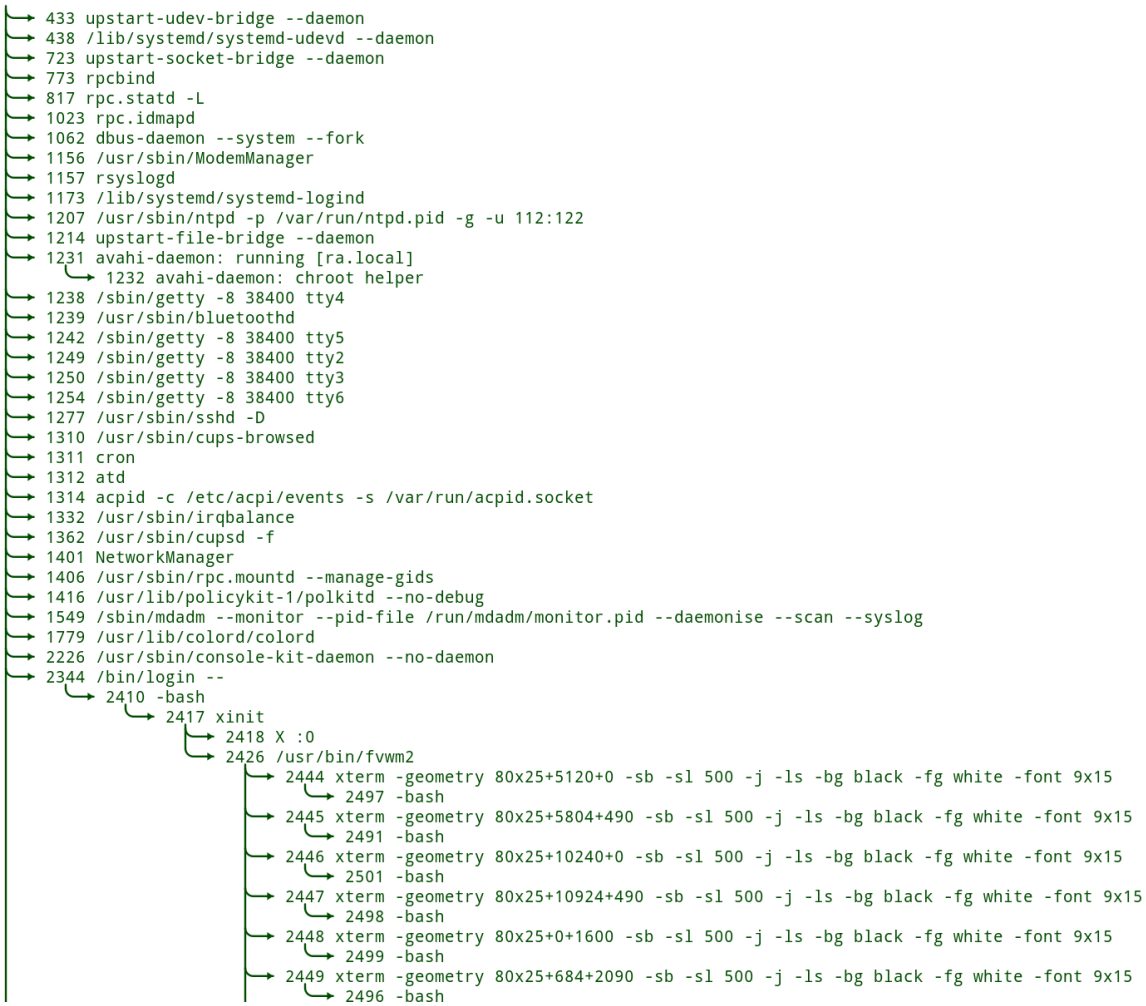
Problem 3

Using the class `LinuxProcess` and `LinuxProcList`, write Python program **`m04p03.py`**, and devise a way to display process tree on the terminal, in a way similar to the ps or pstree commands. Avoid displaying processes which do not have a command line (i.e., pseudo-file /proc/[pid]/cmdline returns empty line).

Problem 4 – bonus mark .2

Using the class `LinuxProcess` and `LinuxProcList`, write Python program **`m04p04.py`**, and devise a way to graphically display process tree on the terminal, in a way similar to the ps or pstree commands. Avoid displaying processes which do not have a command line (i.e., pseudo-file /proc/[pid]/cmdline returns empty line). Use the pycairo module, which wraps the cairo graphics library.

Example of the result:


433 upstart-udev-bridge --daemon
438 /lib/systemd/systemd-udevd --daemon
723 upstart-socket-bridge --daemon
773 rpcbind
817 rpc.statd -L
1023 rpc.idmapd
1062 dbus-daemon --system --fork
1156 /usr/sbin/ModemManager
1157 rsyslogd
1173 /lib/systemd/systemd-logind
1207 /usr/sbin/ntpd -p /var/run/ntpd.pid -g -u 112:122
1214 upstart-file-bridge --daemon
1231 avahi-daemon: running [ra.local]
1232 avahi-daemon: chroot helper
1238 /sbin/getty -8 38400 tty4
1239 /usr/sbin/bluetoothd
1242 /sbin/getty -8 38400 tty5
1249 /sbin/getty -8 38400 tty2
1250 /sbin/getty -8 38400 tty3
1254 /sbin/getty -8 38400 tty6
1277 /usr/sbin/sshd -D
1310 /usr/sbin/cups-browsed
1311 cron
1312 atd
1314 acpid -c /etc/acpi/events -s /var/run/acpid.socket
1332 /usr/sbin/irqbalance
1362 /usr/sbin/cupsd -f
1401 NetworkManager
1406 /usr/sbin/rpc.mountd --manage-gids
1416 /usr/lib/policykit-1/polkitd --no-debug
1549 /sbin/mdadm --monitor --pid-file /run/mdadm/monitor.pid --daemonise --scan --syslog
1779 /usr/lib/colord/colord
2226 /usr/sbin/console-kit-daemon --no-daemon
2344 /bin/login --
2410 -bash
2417 xinit
2418 X :0
2426 /usr/bin/fvwm2
2444 xterm -geometry 80x25+5120+0 -sb -sl 500 -j -ls -bg black -fg white -font 9x15
2497 -bash
2445 xterm -geometry 80x25+5804+490 -sb -sl 500 -j -ls -bg black -fg white -font 9x15
2491 -bash
2446 xterm -geometry 80x25+10240+0 -sb -sl 500 -j -ls -bg black -fg white -font 9x15
2501 -bash
2447 xterm -geometry 80x25+10924+490 -sb -sl 500 -j -ls -bg black -fg white -font 9x15
2498 -bash
2448 xterm -geometry 80x25+0+1600 -sb -sl 500 -j -ls -bg black -fg white -font 9x15
2499 -bash
2449 xterm -geometry 80x25+684+2090 -sb -sl 500 -j -ls -bg black -fg white -font 9x15
2496 -bash