

Python Programming III

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 2](#_Toc476902716)

[Background Reading 2](#_Toc476902717)

[Important Information 2](#_Toc476902718)

[Introduction 3](#_Toc476902719)

[Problem 1 3](#_Toc476902720)

[Problem 2 3](#_Toc476902721)

[Problem 3 4](#_Toc476902722)

Offensive and Defensive Tool Construction

Python Programming III

Objectives

This lab focuses on the following objectives:

* Define a Python class.
* Create a Python object.
* Use class functions.
* Define class methods.
* Explain inheritance.

Background Reading

Read chapters 12–14 in *How to Think Like a Computer Scientist: Learning with Python*, available at [www.greenteapress.com/thinkpython/thinkCSpy.pdf](http://www.greenteapress.com/thinkpython/thinkCSpy.pdf).

# Important Information

* For *every* lab and home assignment, store all your work in your personal repository in a subdirectory named **mXX**, where XX is the module number. Carefully name the program as described in each problem.
* Your programs are extracted from your repository by a Python script. If there are any errors in the program name, then your instructor will never see your program, and you will receive a mark of zero.
* Push your work to the server often, and ensure that you push the final version of a program by the deadline specified, because the script extracting them can be run at any time after the deadline.

# Introduction

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 use the Python graphics library PIL and generate a picture. Then, we will design a simple graphical user interface (GUI) to interact with the application, using GTK3 and the Glade GUI designer.

# Problem 1

Write a Python program named **m04p01.py** that creates the LinuxProcessclass. Write methods to extract the fields, apply the methods and print the results using 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

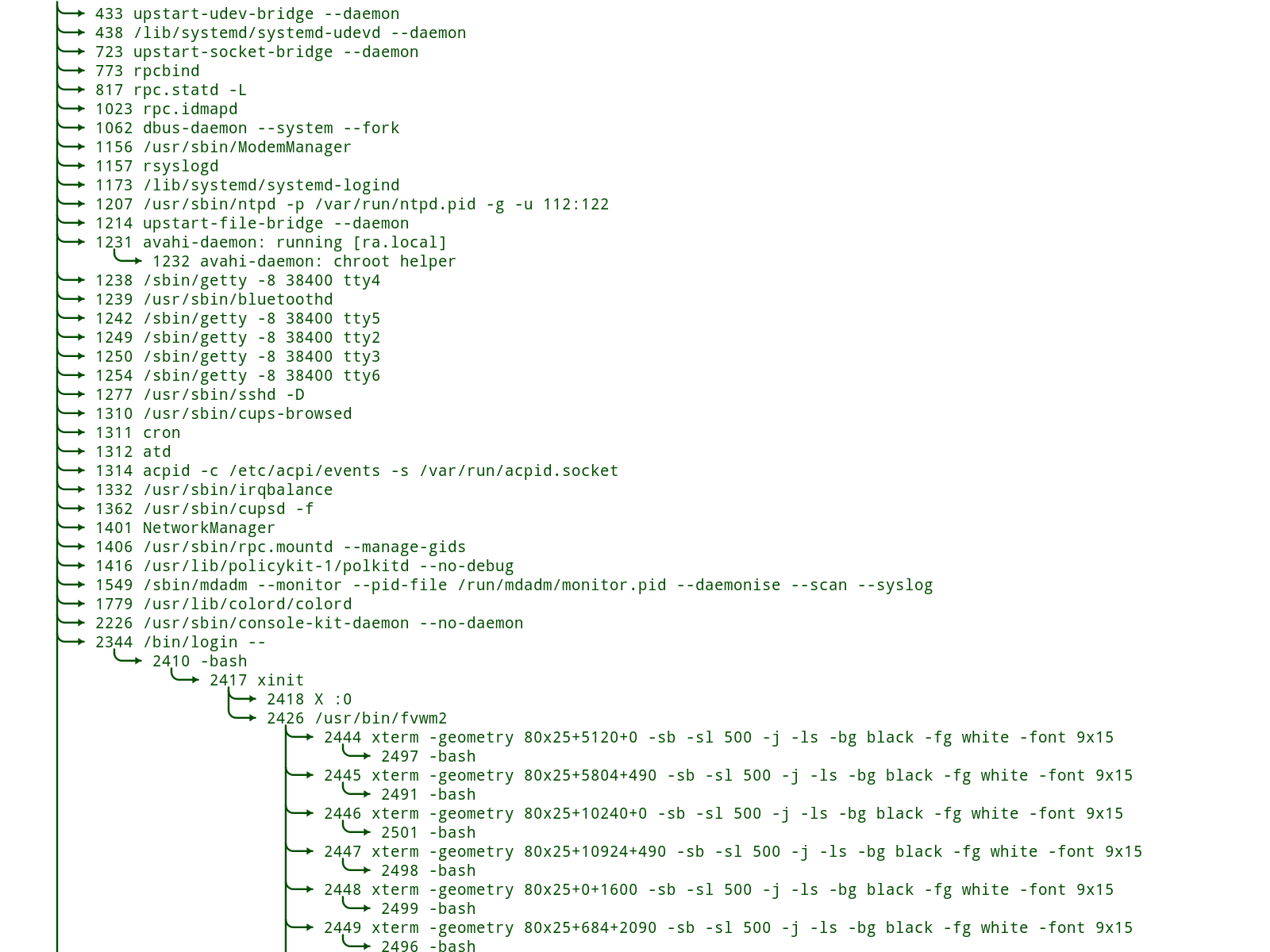
# Problem 2

Write a Python program named **m04p02.py** that creates the LinuxProcList class. This class reads all the running processes from */proc/[pid]/stat* and generates an internal structure that represents the process tree. It also reads 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 function such that:

* LinuxProcList.proclist() returns a list of all process IDs,
* LinuxProcList.cmdline(pid) returns either a string containing the command line for the given process or *None*, and
* LinuxProcList.children(pid) returns a list of the children of the given process.

# Problem 3

Using the LinuxProcess and LinuxProcList classes, write a Python program named **m04p03.py** and devise a way to graphically display a process tree on the terminal, similar to the *ps* or *pstree* commands. Avoid displaying processes that do not have a command line (e.g., the pseudo-file */proc/[pid]/cmdline* returns an empty line). Use the Pycairo module, which wraps the Cairo graphics library.

The following is an example of the result:

**© 2017, Linux**  
Reproduced and used in accordance with SAIT’s Fair Dealing Policy (Schedule A, *Copyright of External Materials Procedure AC.2.12.1*); further distribution may infringe copyright.

**Note:** The rest of the problems for this module are available in the homework assignment. See your course schedule for details.