System Programming with C++

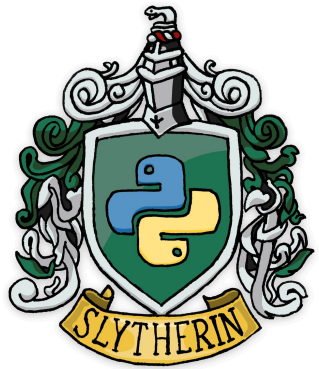
Ideas for projects

1. Python without GIL

1. Python without GIL

Problem: multithreading in Python doesn't work.

Reason: GIL - global interpreter lock.

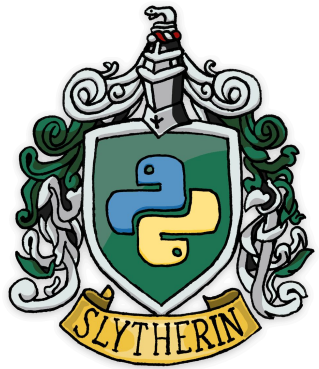


1. Python without GIL

Problem: multithreading in Python doesn't work.

Reason: GIL - global interpreter lock.

Why to have GIL? Because memory manager is based on reference counting and it is not thread safe.



1. Python without GIL

Problem: multithreading in Python doesn't work.

Reason: GIL - global interpreter lock.

Why to have GIL? Because memory manager is based on reference counting and it is not thread safe.

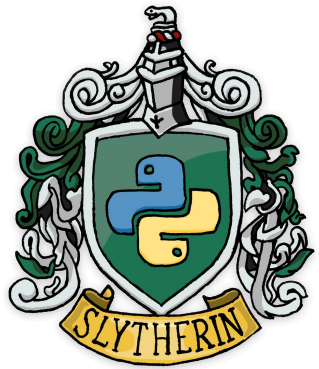
Idea for project: reimplement memory manager for Python. Make your own tracing garbage collector!

1. Python without GIL

Problem: multithreading in Python doesn't work.

Reason: GIL - global interpreter lock.

Why to have GIL? Because memory manager is based on reference counting and it is not thread safe.



Idea for project: reimplement memory manager for Python. Make your own tracing garbage collector!

Details: your own interpreter or some external one;

1. Python without GIL

Problem: multithreading in Python doesn't work.

Reason: GIL - global interpreter lock.

Why to have GIL? Because memory manager is based on reference counting and it is not thread safe.

Idea for project: reimplement memory manager for Python. Make your own tracing garbage collector!

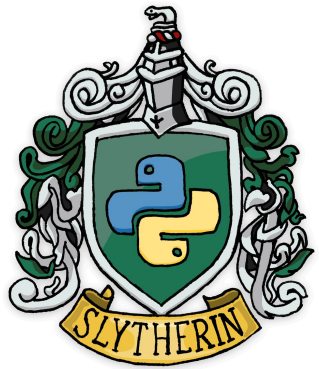
Details: your own interpreter or some external one; CPython based runtime doesn't look like a good idea, but who knows?

1. Python without GIL

Problem: multithreading in Python doesn't work.

Reason: GIL - global interpreter lock.

Why to have GIL? Because memory manager is based on reference counting and it is not thread safe.



Idea for project: reimplement memory manager for Python. Make your own tracing garbage collector!

Details: your own interpreter or some external one; CPython based runtime doesn't look like a good idea, but who knows? No full implementation needed.

2. Your own file system (FUSE)

Idea: implement your own virtual file system in user space.

2. Your own file system (FUSE)

Idea: implement your own virtual file system in user space.

Example (basic): have a single file on the disk and API to work with it as with FS (create/remove folders and files, read and write them, etc)

2. Your own file system (FUSE)

Idea: implement your own virtual file system in user space.

Example (basic): have a single file on the disk and API to work with it as with FS (create/remove folders and files, read and write them, etc)

Example (advanced): the file can be encrypted or heavily compressed.

Example (advanced): the "file" can be located on several machines (distributed FS).

2. Your own file system (FUSE)

Idea: implement your own virtual file system in user space.

Example (basic): have a single file on the disk and API to work with it as with FS (create/remove folders and files, read and write them, etc)

Example (advanced): the file can be encrypted or heavily compressed.

Example (advanced): the "file" can be located on several machines (distributed FS).

3. Your own regular expressions engine

Idea: implement your own regexp library.

3. Your own regular expressions engine

Idea: implement your own regexp library.

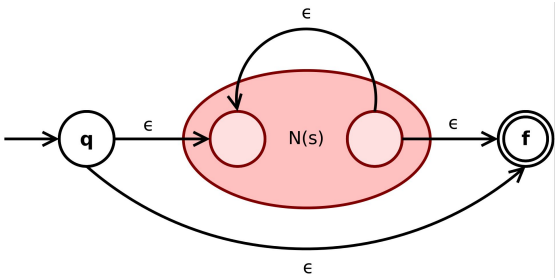
Main feature: optimizations of expression before building an automata

3. Your own regular expressions engine

Idea: implement your own regexp library.

Main feature: optimizations of expression before building an automata

Find some cases where your optimizations will rock and it will give you better performance than usual C++ engine.



4. Your own text-editor/file manager

Idea: implement your own vim or emacs that will work well with big files and look good (via pseudo graphics)

4. Your own text-editor/file manager

Idea: implement your own vim or emacs that will work well with big files and look good (via pseudo graphics)

Main features:

- ide like navigation and actions,
- working with binary files (sections/disassembly)

5. Honorable mentions

1. Your own graphic editor.
2. Your own window manager for Linux.

5. Honorable mentions

1. Your own graphic editor.
2. Your own window manager for Linux.
3. Interpreter of C++ language itself!

5. Honorable mentions

1. Your own graphic editor.
2. Your own window manager for Linux.
3. Interpreter of C++ language itself!
4. An emulator for RISC-V assembly language (your own QEMU).

5. Honorable mentions

1. Your own graphic editor.
2. Your own window manager for Linux.
3. Interpreter of C++ language itself!
4. An emulator for RISC-V assembly language (your own QEMU).
5. Your own archiver with non-standard compression algorithm.

Your ideas?