TroubleShooting is when we're fixing problems in the system running the application, and debugging when we're fixing the bugs in the actual code of the application.

# Week1:

strace ./py\_script.py trace the system calls made by the program & tell us what the result of each of the call was.

strace -o error\_file.strace ./py\_script.py this will store the output into the txt file error\_file.strace

less error\_file.strace this allows you to scroll through the contents of a text file

## UNIX COMMANDS:

top shows the state of the computer & processes using the most CPU.

In the first line this gives us the load\_average. This shows how much time a processor is busy in a given minute.

free shows free memory

w shows the users logged into the system

uptime since how long has the system/server been running

cat /proc/cpuinfo shows number of cpu cores etc.

cat /proc/loadavg since top command itself consumes memory, use this to find the load\_avg. gives load\_avg for 1, 5 and 15 minutes respectively.

# Week2:

How computers use resources?

1. CPU Internal Memory
2. RAM
3. Disk
4. Network

A system tries to store data over CACHE (within CPU Internal Memory) for quick retrieval than go and fetch it from Disk or Network every time it needs it.

SWAP- A space on the Hard Drive where the OS will keep files that it cannot Cache (due to not enough available space).

Retrieval from SWAP (which is on the HD) is slower.

Memory Leak: When a Program that’s not needed now takes all the memory (ideally the memory should be released if the program is not needed now) that is available slowing down other applications. So, Memory that is no longer needed is not getting released.

ab -n 500 www.google.com/ 🡪 this is a way to test the speed of a website. Look at the parameter Time Per request.

The process priorities in Linux is denoted by a number, the smaller the number the higher the priority. 0 to 19. By default, processes start with a priority of 0. We can change them using the function nice and renice.

pidof name\_of\_process 🡪 gives the process id of a process.

for pid in $(pidof process\_name); do renice 19 $pid; done

ps ax 🡪 shows all the currently running programs on the System

killall -STOP process\_name 🡪 this will stop the process but not kill it.

locate filename 🡪 to locate the folder where a file exists.

Profiler: A tool that measures the resources that our code is using, giving us a better understanding of what’s going on. Tell us how the memory is allocated and how time is spent. cProfile module in Python. The cProfile module is used to count how many times functions are called, and how long they run.

pprofile3 –f callgrind –o profile.out ./py\_file.py 🡪 -f will tell it to use the callgrind format & -o to tell it to store the output in profile.out file. To open the profile.out file use kcachegrind profile.out

<https://realpython.com/python-concurrency/>

<https://hackernoon.com/threaded-asynchronous-magic-and-how-to-wield-it-bba9ed602c32>

## CPU bound:

CPU bound means the program is bottlenecked by the CPU (Central Processing Unit). When your program is waiting for I/O (e.g., disk read/write, network read/write), the CPU is free to do other tasks, even if your program is stopped. The speed of your program will mostly depend on how fast that I/O can happen; if you want to speed it up, you'll need to speed up the I/O. If your program is running lots of program instructions and not waiting for I/O, then it's CPU bound. Speeding up the CPU will make the program run faster.

In either case, the key to speeding up the program might not be to speed up the hardware but to optimize the program to reduce the amount of I/O or CPU it needs. Or you can have it do I/O while it also does CPU-intensive work. CPU bound implies that upgrading the CPU or optimizing code will improve the overall computing performance.

rsync [Options] [Source-Files-Dir] [Destination]

Some of the commonly used options in rsync command are listed below:

|  |  |
| --- | --- |
| **Options** | **Uses** |
| -v | Verbose output |
| -q | Suppress message output |
| -a | Archive files and directory while synchronizing |
| -r | Sync files and directories recursively |
| -b | Take the backup during synchronization |
| -z | Compress file data during the transfer |

Example:

1. Copy or sync files locally:

rsync -zvh [Source-Files-Dir] [Destination]

1. Copy or sync directory locally:

rsync -zavh [Source-Files-Dir] [Destination]

1. Copy files and directories recursively locally:

rsync -zrvh [Source-Files-Dir] [Destination]

To learn more about rsync basic command, check out [this link](https://www.linuxtechi.com/rsync-command-examples-linux/).

os.walk returns a generator, that creates a tuple of values (current\_path, directories in current\_path, files in current\_path)