Our first administrative command - top

At this moment we know a lot of commands. We are ready to look on some administrative side of the work with system. We will learn how to take a basic look on it. But don't be fooled, basic doesn't mean this command is very simple. It is not. The data collected is very vast and informative.

top

We talk here about top command. Let's execute it and then we will go through the displayed information line by line.

top

First line

top - 19:38:28 up 2 days, 20:47, 0 users, load average: 0.52, 0.58, 0.59

In the first line we see something similar to the example above. Let's go through it one by one.

top - program name  
19:38:28 - current hour, obvious :)  
up 2 days, 20:47 - uptime. Another words, the time from last start of the system.  
0 users - number of *active* users. Here we can see similar information, like with command who . Let's try. First, we need to quit the top:

q

and now we can run who command:

who

The main purpose of who is to show who is logged in. We will touch this command in the future labs.

Let's come back to top screen.

top

The last part, the load average, is very important, yet very often misunderstood.

load average: 0.52, 0.58, 0.59

Let's go through it.

We see here three numbers. They are representing the load average for the system in last 1, 5 and 15 minutes. These shows the average number of processes **running** and **waiting for CPU** time.

It is **crucial** to understand, that these values need to be evaluated very closely with the number of CPUs, Cores, Threads. The number 10 means massive overload when your system has 1 core, but is quite ok when your system has 12 cores. We will learn how to see number of cores in top a little bit later. This is the most basic explanation of load average, and please, be sure, you understand it.

This first line is exactly the same like we have in w command. Let's see.

q for exit the top command,

w

This command shows logged users too, but the first line is exactly the same like in top.

Ok, let's come back to our top command.

top

Second line

Tasks: 6 total, 1 running, 5 sleeping, 0 stopped, 0 zombie

Second line shows us information about processes in our system. What every type means?

* total - shows all processes in the system
* running - currently active processes. It means, these processes are using CPU right now
* sleeping - generally - process is waiting for something. It may be I/O operation for example.
* stopped - Stopped processes (for example by ctrl+z)
* zombie - Very important state to understand. It is a process which had finish his job but still has entry in the process table. In simple way, these processes are waiting for exit(). It may happen, when parent process deteriorated somehow. Sometimes we are able to kill zombie (by killing the parrent), but in may cases it will not work. But it is not a place to talk about it :)

You will notice very shortly, that total doesn't repesent all processes. For example, you will not find idle state here.

Third line

So far so good. Now it is time for the third line.

%Cpu(s): 13.9 us, 9.5 sy, 0.0 ni, 76.3 id, 0.0 wa, 0.4 hi, 0.0 si, 0.0 st

This line shows the CPU(s) utilization, splitted to specific types. Let's go through them one by one.

* us - user - All user processes are combined in this number. So, our sessions too.
* sy - system - processes owned by system (kernel)
* ni - nice - this is important to understand. nice allows us to change the priority of the process. The standard value for processes is 0 , but we can modify it from 19 (lowest) to -20 (highest) priority. This statistic here shows all processes with the niceness set abow 0. So, the processes which will be executed by the system, when "systemm will have time for it".
* id - idle - idle time means that the system is bored and do nothing.
* wa - iowait - the number repspresents the time (which is a subset of idle time) when the process is waiting for input/output operation. This statistic is very important, because it may show the issue outside the CPU, in other hardware (but not only) components.
* hi - hardware interrupts. These are physical interrrupts from hardware and are handled by CPU itself.
* si - software interrupts. These are generated by software and are handled by kernel.
* st - steal time - very important to understand, especially when we are working on virtualized environment. This number represents the time "stealed" from the virtual machine by hypervisor. Another words, how long our system needs to wait for resources from hypervisor.

Fourth and fifth lines

MiB Mem : 16217.5 total, 6184.9 free, 9808.7 used, 224.0 buff/cache

MiB Swap: 49152.0 total, 48436.2 free, 715.8 used. 6278.3 avail Mem

We will go through these two lines together, as both represent the memory information. The only one difference is that the first line is about physical memory and second is about swap. We will talk about swap in future lesson.

total, free and used is obvious.

buff/cache is a combine value of *buffer* memory, used by kernel and *cache*, memory by page cache.

available simply means that the new starting program, application, etc can use max this size of memory for its to be run.

Processes list

Below these five lines we have processes list. This list contains fields, so let's go through them to better understand the meaning.

* PID - Process ID number. It is unique number of the process in the system.
* USER - process' owner. The process is started by this user.
* PR - default priority of the process, scheduled by kernel when process was started.
* NI - nice. Shows the value, if nice was performed against the process.
* VIRT - total amount of memory used by the process.
* RES - RAM memory used by process.
* SHR - amount of memory shared with other processes.
* S - process state (we discussed it above).
* %CPU - what amount of available CPU is used by the process.
* %MEM - like for CPU, but this value represents memory usage.
* TIME+ - total time of CPU usage by the process.
* COMMAND - quite obvious, this process is executed.

# How to modify the default view

top is a very powerful tool. The main functionality we already know, but it is not all. First, we can modify a lot of visual aspects. Let's do it!

## CPU and memory list

Hit 1

And again 1

And again 1 .

What we see here? The CPU information in line 3 is expanded to single cores. We switch here between the unified view and detailed view.

With Killercoda's default setting we probably see only one core. Please, be observant, and notice the difference between Cpu(s) and Cpu0.

Press t couple of times and then

m , also couple of times. Observe, how the CPU or memory visualisation is changing.

## Modify the order in the processes list

Now it is time to modify what we see in processes list. Be sure that you have top running.`

The standard view is by CPU usage.

Let's switch to order by memory usage. In order to do it, press

M .

Now, by pressing

N we will look on the list sorted by PID.

Ok, Let's press

T and now we have order by running time.

Finally we can come back to CPU usage sort, by pressing

P

## Tasks and threads

By default our processes list contains the tasks. We can switch it to threads by pressing H (and come back to processes by pressing the same key).

## Paths and trees

By pressing c we can change between simple name of the command and the full path.

This might be useful with v and V. Please combine them (v, V and H) and play a little. v and V shows the tree of processes, instead simple list. This might help to understand the parent / child relations between processes.

## Filtering

Press

n

in order to limit the number of shown processes / threads. When 0 is provided, it means simply all processes.

The prompt for our action appears just above the process list.

We can confirm our selection, by pressing enter or reject it, by using ESC.

When

u

is used, we can select one user to list his processes.

With

o

or

O

we can search through processes. o is case-insensitive and O is case-sensitive.

## Refresh

With s we can change the refresh rate. The default is 3 seconds. We may set the refresh as slower or faster.

## Kill the process

We didn't talk about it, yet. For now, it is enough to say, we can kill the process using top. In order to do so, press k and provide the PID of the process to kill.

## Save the changes

Now interesting thing. If many settings were changed and we wish to keep them for longer time, we don't need to provide all of them again when we use top again. Simply press

W

and the settings will be written in the toprc file in the home directory (sometimes the file can be just .toprc in your home).

Let's see this file.

cat .config/procps/toprc

And now run the top again.

We touched here very limited number of options. There is much, much more to explore. Feel free to go through manual and try by yourself. To learn more, you can read manual

# Start with non default settings

Many of options we used before can be executed directly from the command line. Here are the examples:

top -o %MEM - run top with processes sorted by memory.

q for exit

top -c will show the full paths.

q for exit, again.

And the last example:

top -u root will show the processes owned by root user.