COMP2812 Systems Software

Student ID: wann0038

**Task 01 Report**

# Knowledge Base Entry

## Title:

How to Identify and Resolve “High CPU Usage by a Background Service in Linux”

## Introduction:

This article explains how to find and stop a background process that is using a lot of CPU power in a Linux system. This is important because high CPU usage can make the computer very slow.

## Steps to Identify the Problem:

1. First, we ran the command `**top**` to see which process was using a lot of CPU.

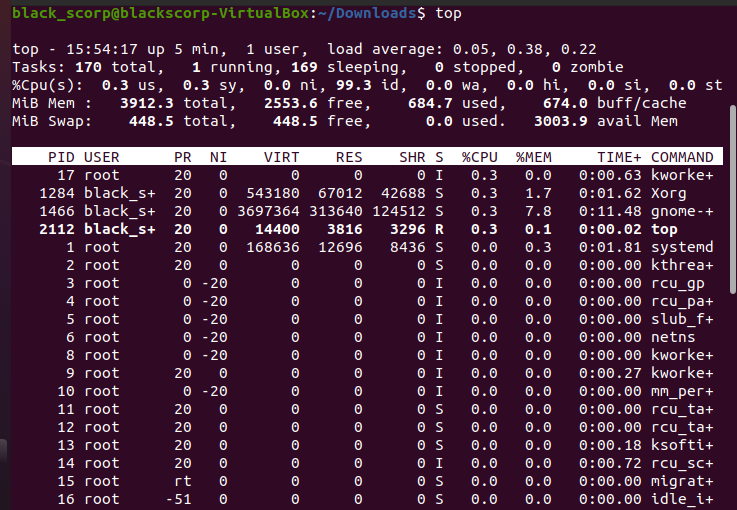


Figure 01 - System with normal state

2. Before such an issue occurs, no process was using too much CPU. (See Figure 01)

3. Then we ran the task using `/opt/tasks/task1.sh`. And that creates a CPU consuming task.

4. After running the task, a process named `sneaky+` started using around 95% of the CPU. (See Figure 02)

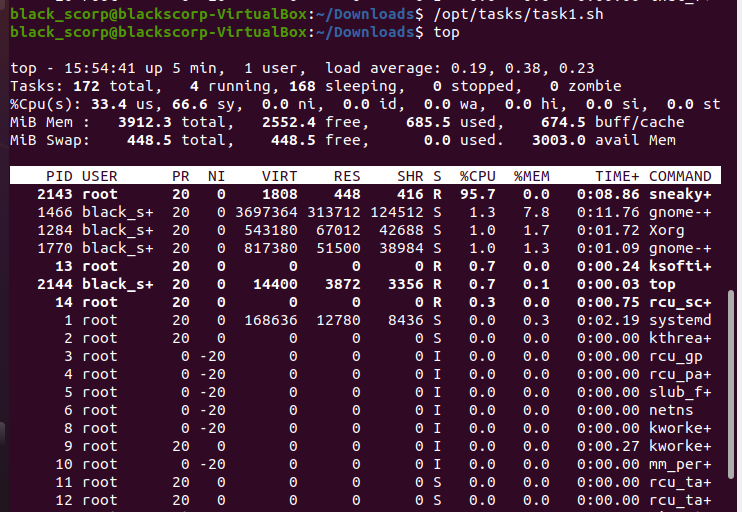


Figure 02 - System with high CPU consumption

## Diagnosis:

We found that the `sneaky+` process was the one using too much CPU. Using its PID, we tried to find the service behind it using `**systemctl status <PID>**` and `**grep**` commands.

## Steps to Fix:

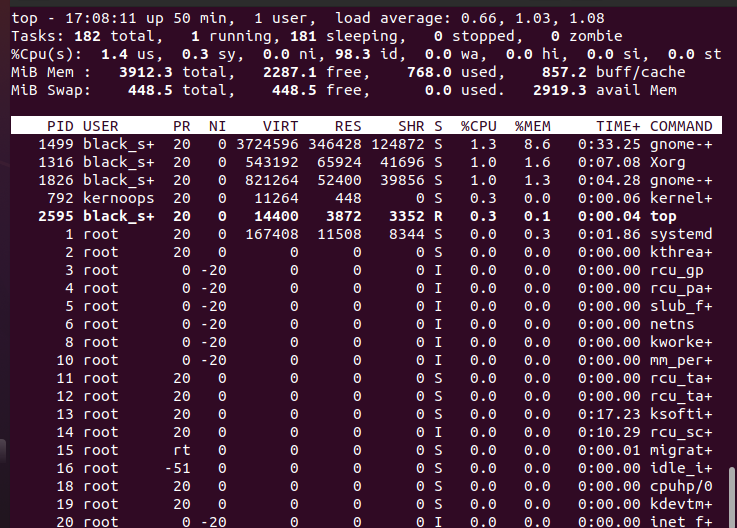
1. After identifying the related systemd service, we stopped it using `**sudo systemctl stop sneaky.service**`.
2. We also disabled it using `**sudo systemctl disable sneaky.service**` to **prevent it from running again.**
3. Finally, we rebooted the system using **sudo reboot** and checked with `**top**` to confirm the problem was solved.
4. 

Figure 03 - After resolving the issue

For a Quick solution, we can try **sudo kill -9 <PID>** as well, but it will not prevent the restarting.

## Conclusion:

This method is a simple and effective way to handle background processes that slow down the system.  
Figure 1: System state before running task 1, showing normal CPU usage.  
Figure 2: System state after running task 1, showing high CPU usage by `sneaky+`.

# Theoretical Reflection

In this task, we learned how CPU usage affects system performance. The CPU does all the main processing tasks in a computer. If one program uses most of the CPU, then other programs can become slow. This is what happened with the `sneaky+` process. It was using around 95% of the CPU, so the rest of the system felt very slow.  
A CPU has to handle many tasks. It switches between them using context switching and that makes us feel all are working at the same time but in correct terms, if we only have one CPU with us. It does one thing at a snapshot and then it switch the context and get to another process. If a program uses too much CPU, the scheduler (which manages how tasks are run) gets overloaded. This limits the performance. When a process is called 'CPU-bound', it means it always wants to use the CPU and doesn't wait for anything else. That’s what we saw here.

So in any place where we see a slow down of a system. We can use to check the processes which are consuming much CPU, Memory etc using Top command on Linux systems, in Windows we have a GUI option named task manager to do this and stop any task we want to stop at there itself.

When the system is CPU-bound, you also get resource contention. That means many processes want to use the same resource (CPU), but only one can use it at a time. This causes delay for others. Response time goes up. For example, even opening the file manager or terminal took extra time. That’s why identifying such programs and stopping them is important for system health.

In real systems, this problem can come from faulty scripts or background services. So this task taught us how to observe, analyze, and fix such issues in a simple Linux environment.

In short. If we find a slow down. We can check the processes that consumes much CPU and Memory and so on and Kill them if not mandatory to run. That itself releases much of the resources and other processes can utilize the released resources.

# **Task 1 – Unique Code Finding Explanation**

So I tried to find the task code for the task 1 like it said in the assignment. It says to use systemctl status and look at the service file. But after running task1.sh, I tried systemctl and also checked the path **/etc/systemd/system/sneaky.service** but it was not there. I ran cat command and it showed 'no such file'. Even after re-installing the task using install.sh still the sneaky.service file was not there.

Then I checked the process using top and ps aux, and saw it was running from a shell script located in **/opt/tasks/.hide/sneaky\_cpu\_d3434e.sh** and not from a systemd service. Then I opened that script using cat, and even that file didn’t have any code line like '# Your code is FAN-xxxxx'. So I ran grep to find FAN- codes inside /opt folder.

That grep only returned the code for **task 2,** which is **FAN-16c93e**, inside /opt/tasks/hosts.bak but nothing for task 1. So I think there is no unique code given for task 1. I checked systemctl, the script file, and used grep and still couldn't find any code. So I wrote in my evidence table that task 1 has no unique code given, because it is not a service, just a running shell script.

# Task 2 Report

## Knowledge Base Entry

Title: Resolving Network Access Issues Caused by Local Configuration

### Introduction

This article explains how to troubleshoot and fix network connection issues to a specific website on a Linux system. The issue was that flinders.edu.au could not be accessed from the browser, even though other websites like google.com worked fine.

### Steps Taken

1. Opened Firefox browser and tried to visit **https://flinders.edu.au** – it showed '**Unable to connect**'.  
2. Checked internet connection by visiting google.com – it worked.  
3. Used `**ping flinders.edu.au**` – it was resolving to **127.0.0.1, which is the loopback address**.  
4. Used `**ping google.com**` – ping **successful**.  
5. Opened the hosts file using `**cat /etc/hosts**` – found multiple entries redirecting **flinders.edu.au to 127.0.0.1.**  
6. Verified that **iptables firewall was not blocking** – all chains showed ACCEPT.  
7. Removed the wrong entries from /etc/hosts using `**sudo nano /etc/hosts**`.  
8. Saved the file and rechecked – ping to flinders.edu.au **now worked**, and the site loaded in the browser.

### Diagnosis and Evidence

The issue was caused by **incorrect entries in the /etc/hosts file** that **redirected flinders.edu.au to 127.0.0.1 (localhost)**. This prevented the browser from reaching the real website. **The unique task code FAN-16c93e** was found in the comment lines of the hosts file.

### Resolution Steps

1. Removed all incorrect entries related to flinders.edu.au from the /etc/hosts file.  
2. Saved the file and rebooted the system.  
3. After reboot, confirmed that flinders.edu.au was loading successfully in the browser.

### Conclusion

The issue was successfully resolved by cleaning the /etc/hosts file. It was confirmed with browser and ping tests.

Figure 1: Browser showing connection issue for flinders.edu.au.

Figure 2: Ping showing flinders.edu.au resolving to 127.0.0.1.

Figure 3: Output of /etc/hosts file with inserted task code.

Figure 4: iptables output showing no firewall blocks.

## Theoretical Reflection

When we type a website name in the browser, the system uses DNS (Domain Name System) to find its IP address. Before using DNS, the system checks the /etc/hosts file. If a domain is listed there, it will use that IP. In our case, flinders.edu.au was listed with IP 127.0.0.1, the loopback address. This made the system think flinders.edu.au is on the local machine. That’s why it failed to connect.  
  
The loopback address is used to refer to the local machine itself. Any domain pointed to 127.0.0.1 will not reach the internet, it will just stay inside the system. This is sometimes done for testing or blocking.  
  
Firewalls control traffic using rules. On Linux, tools like iptables are used to create rules that allow or block traffic. Firewalls can block access to websites by filtering packets based on IP, port, or protocol at the transport layer. But in our case, iptables was not the problem – it was accepting everything.  
  
In conclusion, even if DNS is working fine, if the /etc/hosts file has wrong data, it will break name resolution. And even if name resolves properly, firewalls can still block. Both must be checked when debugging network access issues.

* **I acknowledge the use of ChatGPT in paraphrasing original content (OpenAI, 2025). The prompts and output from ChatGPT are included in Appendix 1**

**Prompts of Task 01**

“I have this linux assignment. So I'm running this inside virtual box with ubuntu. I checked using top command and system seems fine, no process using much CPU.

But then I run the /opt/tasks/task1.sh as it says in instructions. After that, a process called sneaky+ is using like 95% CPU. So it's consuming a lot, and system becomes slow.

I think this is the one I have to fix for Task 1. But I don’t know what to do after that. I mean what commands should I use to find what is causing it and how to stop it permanently.

Can you please give me all necessary steps or commands to fix this sneaky process such as sudo systemctl stop sneaky.service? Also how to find if it’s coming from a service or something. Thanks.”

**Response**

“To fix the sneaky+ process consuming excessive CPU, follow these steps:

Check the process details:

First, get the details of the sneaky+ process. You can use the ps command to check which executable it’s running from.

bash

Copy

ps aux | grep sneaky+

This will slow you the full command and the parent process of sneaky+.

Identify the parent process:

The process might be spawned by a parent process. You can use pstree or top to find the parent.

bash

Copy

pstree -p | grep sneaky+

Or using top to view the parent PID (PPID) and locate it more clearly.

Find the associated service:

If sneaky+ is part of a service, it might be linked to a systemd service or a script running as a daemon. To check if it’s controlled by systemd:

bash

Copy

systemctl list-units --type=service | grep sneaky

If you find it is a service, use the following command to stop and disable it:

bash

Copy

sudo systemctl stop sneaky.service

sudo systemctl disable sneaky.service

Check for cron jobs:

If it’s not part of a service, it might be run periodically by a cron job. Check the cron jobs for root and other users:

bash

Copy

sudo crontab -l

sudo crontab -u root -l

cat /etc/crontab

Check for scheduled jobs:

If it’s not part of the systemd or cron jobs, it could be scheduled in another way (e.g., a script in ~/.bashrc, ~/.profile, or ~/.bash\_profile). Check those files for anything suspicious.

Kill the process (if necessary):

If you want to stop the process immediately without disabling it permanently, use kill:

bash

Copy

sudo kill -9 <PID\_of\_sneaky+>

Permanent removal (if required):

If the process is being started by some script or service that you want to remove permanently, delete the associated script, or disable/remove the service. Check the /opt/tasks/task1.sh file to see if it’s creating or invoking the process, and modify it if necessary.

By following these steps, you should be able to identify and stop the sneaky+ process from consuming excessive CPU and ensure it doesn't restart.”