

# CS310 PROJECT

MINI TASK MANAGER FOR  
LINUX

---

Sayanth Sunil  
2303133

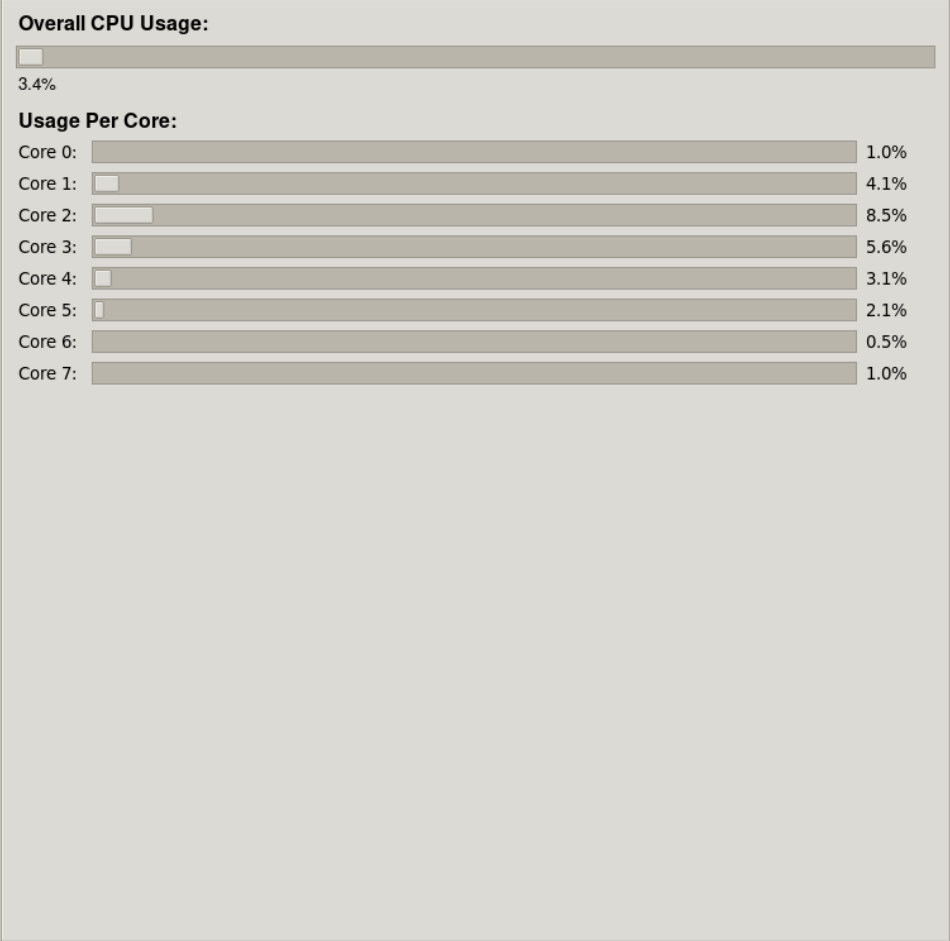
**Instructions :** since it works using /proc file system ,this will only work on linux operating system and on WSL



# USER INTERFACE

# FULL VIEW

## CPU and Core Monitoring



## Memory Monitoring



## Process Monitoring

All Processes

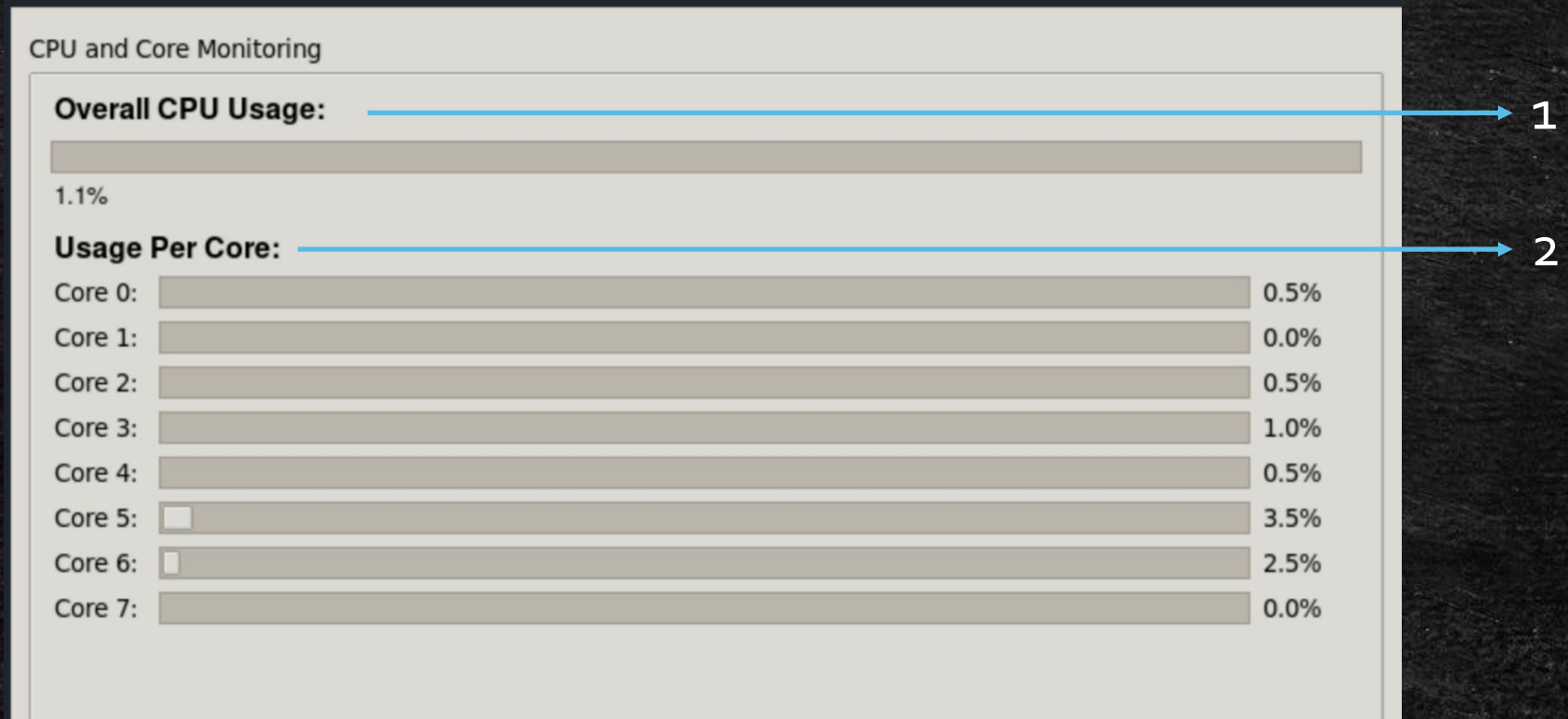
User Processes

System Processes

Pid	User	Cpu	Mem	Status	Command
833	root	2050.0	157.3M	S (sleeping)	/usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run/lightdm/r
16958	1000	850.0	40.2M	S (sleeping)	xfce4-screenshooter
1166	1000	450.0	122.6M	S (sleeping)	xfwm4 --display :0.0 --sm-client-id 20eff90d6-b87a-44e
4840	1000	450.0	24.7M	R (running)	python3 23.py
31	root	50.0	0.0M	S (sleeping)	migration/2
799	root	50.0	2.5M	S (sleeping)	/usr/sbin/VBoxService
1102	1000	50.0	3.0M	S (sleeping)	/usr/bin/VBoxClient --draganddrop
1232	1000	50.0	87.6M	S (sleeping)	/usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/l
1	root	0.0	12.8M	S (sleeping)	/sbin/init splash
2	root	0.0	0.0M	S (sleeping)	kthreadd
3	root	0.0	0.0M	S (sleeping)	pool_workqueue_release
4	root	0.0	0.0M	I (idle)	kworker/R-rcu_gp
5	root	0.0	0.0M	I (idle)	kworker/R-sync_wq
6	root	0.0	0.0M	I (idle)	kworker/R-slub_flushwq
7	root	0.0	0.0M	I (idle)	kworker/R-netns
10	root	0.0	0.0M	I (idle)	kworker/0:0H-kblockd
12	root	0.0	0.0M	I (idle)	kworker/R-mm_percpu_wq
13	root	0.0	0.0M	I (idle)	rcu_tasks_kthread
14	root	0.0	0.0M	I (idle)	rcu_tasks_rude_kthread
15	root	0.0	0.0M	I (idle)	rcu_tasks_trace_kthread
16	root	0.0	0.0M	S (sleeping)	ksoftirqd/0
17	root	0.0	0.0M	I (idle)	rcu_preempt
18	root	0.0	0.0M	S (sleeping)	rcu_exp_par_gp_kthread_worker/0
19	root	0.0	0.0M	S (sleeping)	rcu_exp_gp_kthread_worker
20	root	0.0	0.0M	S (sleeping)	migration/0
21	root	0.0	0.0M	S (sleeping)	idle_inject/0



# CPU AND CORE MONITORING PART



1. Overall CPU usage percentage will be showed here
2. Usage percentage of each core will be shown here

# Code

```
def get_system_cpu_times(self):
    """Reads /proc/stat to get total and per-core CPU times."""
    try:
        with open('/proc/stat', 'r') as f:
            lines = f.readlines()

        # Overall is the first line, 'cpu'
        overall_times = [int(p) for p in lines[0].split()[1:]]

        # Per-core lines start with 'cpuX'
        core_times = []
        for i in range(self.cpu_core_count):
            core_line = lines[i + 1]
            core_times.append([int(p) for p in core_line.split()[1:]])
        return {'overall': overall_times, 'cores': core_times}
    except (IOError, ValueError):
        return {'overall': [0]*10, 'cores': [[0]*10 for _ in range(self.cpu_core_count)]}
```



```
def update_cpu_info(self):
    current_sys_cpu_times = self.get_system_cpu_times()

    # --- Overall CPU Calculation Logic ---
    prev_overall = self.prev_sys_cpu_times['overall']
    curr_overall = current_sys_cpu_times['overall']

    prev_total = sum(prev_overall)
    curr_total = sum(curr_overall)
    prev_idle = prev_overall[3]
    curr_idle = curr_overall[3]

    total_delta = curr_total - prev_total
    idle_delta = curr_idle - prev_idle

    cpu_usage = 0.0
    if total_delta > 0:
        cpu_usage = 100.0 * (1.0 - idle_delta / total_delta)

    self.overall_cpu_bar['value'] = cpu_usage
    self.overall_cpu_label['text'] = f"{cpu_usage:.1f}%"
```

```
# --- Per-Core CPU Calculation Logic ---
for i in range(self.cpu_core_count):
    prev_core = self.prev_sys_cpu_times['cores'][i]
    curr_core = current_sys_cpu_times['cores'][i]

    prev_total_core = sum(prev_core)
    curr_total_core = sum(curr_core)
    prev_idle_core = prev_core[3]
    curr_idle_core = curr_core[3]

    total_delta_core = curr_total_core - prev_total_core
    idle_delta_core = curr_idle_core - prev_idle_core

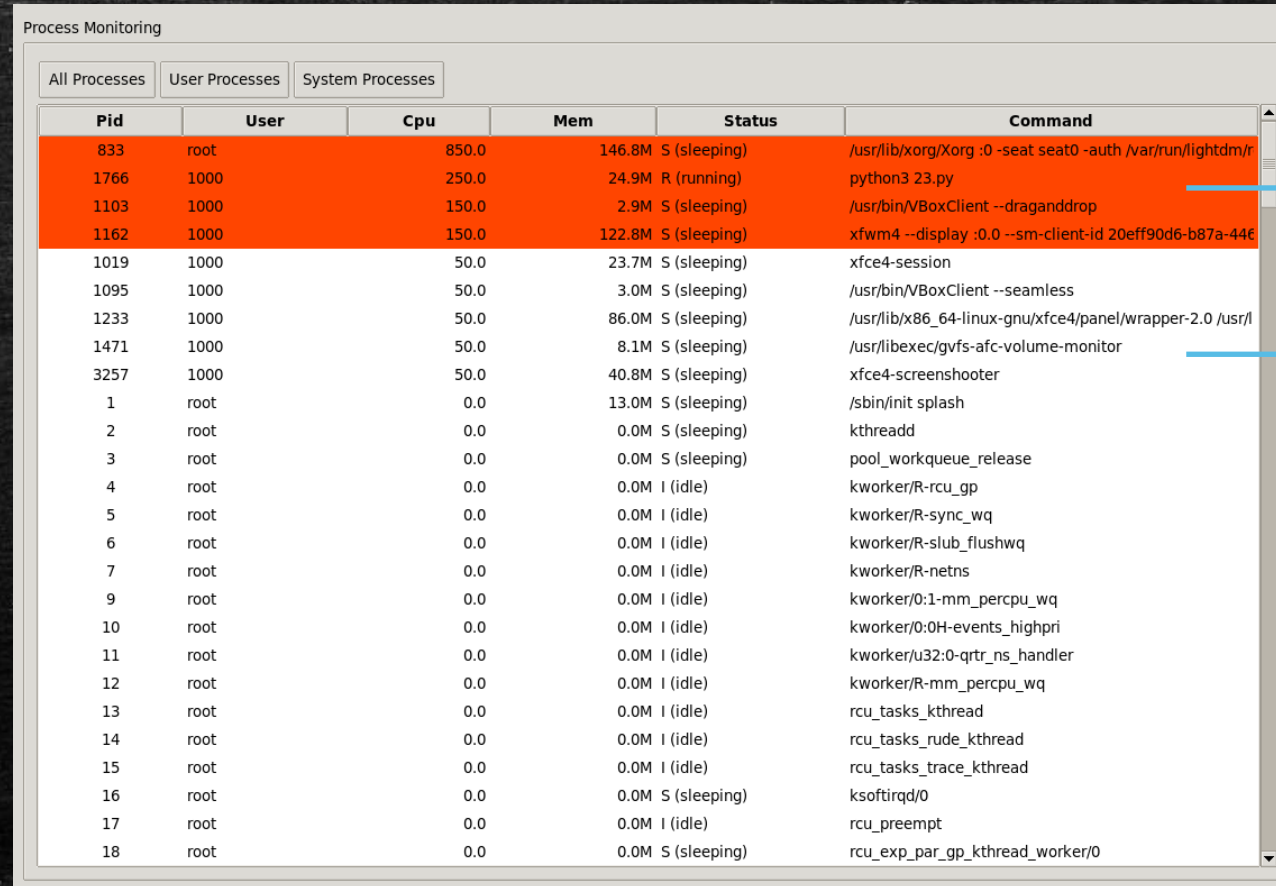
    core_usage = 0.0
    if total_delta_core > 0:
        core_usage = 100.0 * (1.0 - idle_delta_core / total_delta_core)

    self.coreBars[i]['value'] = core_usage
    self.core_labels[i]['text'] = f"{core_usage:.1f}%"

self.prev_sys_cpu_times = current_sys_cpu_times
```



# PROCESS MONITORING PART



Process Monitoring

All Processes User Processes System Processes

Pid	User	Cpu	Mem	Status	Command
833	root	850.0	146.8M	S (sleeping)	/usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run/lightdm/r
1766	1000	250.0	24.9M	R (running)	python3 23.py
1103	1000	150.0	2.9M	S (sleeping)	/usr/bin/VBoxClient --draganddrop
1162	1000	150.0	122.8M	S (sleeping)	xfwm4 --display :0.0 --sm-client-id 20eff90d6-b87a-446
1019	1000	50.0	23.7M	S (sleeping)	xfce4-session
1095	1000	50.0	3.0M	S (sleeping)	/usr/bin/VBoxClient --seamless
1233	1000	50.0	86.0M	S (sleeping)	/usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/l
1471	1000	50.0	8.1M	S (sleeping)	/usr/libexec/gvfs-afc-volume-monitor
3257	1000	50.0	40.8M	S (sleeping)	xfce4-screenshooter
1	root	0.0	13.0M	S (sleeping)	/sbin/init splash
2	root	0.0	0.0M	S (sleeping)	kthreadd
3	root	0.0	0.0M	S (sleeping)	pool_workqueue_release
4	root	0.0	0.0M	I (idle)	kworker/R-rcu_gp
5	root	0.0	0.0M	I (idle)	kworker/R-sync_wq
6	root	0.0	0.0M	I (idle)	kworker/R-slub_flushwq
7	root	0.0	0.0M	I (idle)	kworker/R-netns
9	root	0.0	0.0M	I (idle)	kworker/0:1-mm_percpu_wq
10	root	0.0	0.0M	I (idle)	kworker/0:0H-events_highpri
11	root	0.0	0.0M	I (idle)	kworker/u32:0-qtr_ns_handler
12	root	0.0	0.0M	I (idle)	kworker/R-mm_percpu_wq
13	root	0.0	0.0M	I (idle)	rcu_tasks_kthread
14	root	0.0	0.0M	I (idle)	rcu_tasks_rude_kthread
15	root	0.0	0.0M	I (idle)	rcu_tasks_trace_kthread
16	root	0.0	0.0M	S (sleeping)	ksoftirqd/0
17	root	0.0	0.0M	I (idle)	rcu_preempt
18	root	0.0	0.0M	S (sleeping)	rcu_exp_par_gp_kthread_worker/0

- 1.The processes which exceed the threshold for CPU is highlighted in orange colour and will be listed in the top
- 2.All other processes happening will be listed down

## code

```
def set_filter(self, filter_type):
    self.current_filter = filter_type

def get_process_data(self):
    processes = []
    pids = [pid for pid in os.listdir('/proc') if pid.isdigit()]

    current_proc_times = defaultdict(int)

    # Get total system CPU time delta
    prev_total_sys_time = sum(self.prev_sys_cpu_times['overall'])
    curr_total_sys_time = sum(self.get_system_cpu_times()['overall'])
    sys_time_delta = curr_total_sys_time - prev_total_sys_time
    if sys_time_delta == 0: sys_time_delta = 1
```





```

# Command from /proc/[pid]/cmdline
cmdline = proc_info.get('Name', 'N/A')
with open(f'/proc/{pid}/cmdline', 'r') as f:
    full_cmd = f.read().replace('\0', ' ').strip()
    if full_cmd: cmdline = full_cmd

mem_kb = int(proc_info.get('VmRSS', '0 kB').replace('kB', '').strip())
uid = int(proc_info.get('Uid', '0').split()[0])

proc_data = {
    'pid': pid, 'user': 'root' if uid == 0 else str(uid), 'cpu': f"{cpu_percent:.1f}",
    'mem': f"{mem_kb/1024:.1f}M", 'status': proc_info.get('State', '?'),
    'command': cmdline, 'uid': uid, 'mem_raw': mem_kb, 'cpu_raw': cpu_percent
}
processes.append(proc_data)
except (IOError, IndexError, ValueError):
    continue

self.prev_proc_times = current_proc_times
return processes

```



```
def update_process_info(self):
    # Clear existing entries
    for i in self.tree.get_children():
        self.tree.delete(i)

    processes = self.get_process_data()

    # Apply filter
    if self.current_filter == 'user':
        processes = [p for p in processes if p['uid'] >= USER_UID_MIN]
    elif self.current_filter == 'system':
        processes = [p for p in processes if p['uid'] < USER_UID_MIN]

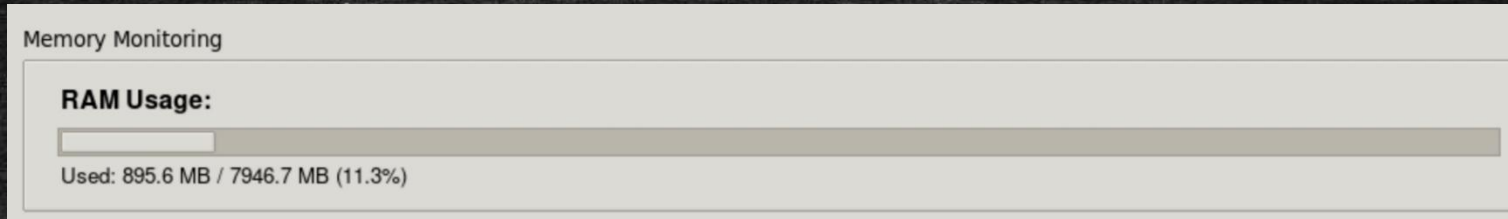
    # Sort by CPU
    processes.sort(key=lambda p: p['cpu_raw'], reverse=True)

    # Populate the treeview
    for proc in processes:
        tags = []
        if proc['cpu_raw'] > CPU_HIGH_THRESHOLD:
            tags.append('high_cpu')
        if proc['mem_raw'] > MEM_HIGH_THRESHOLD_KB:
            tags.append('high_mem')

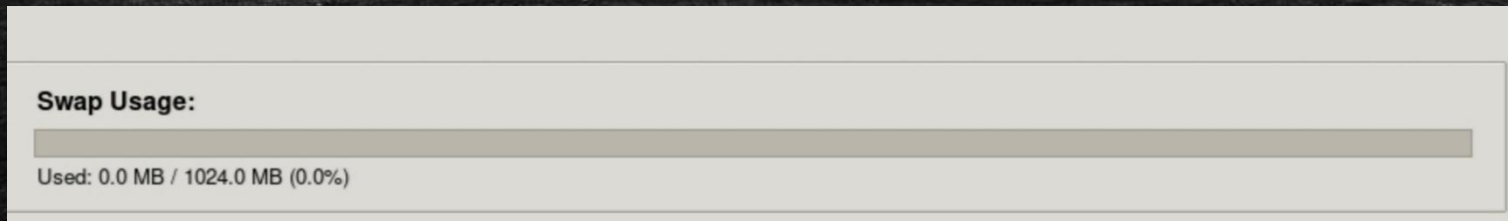
        self.tree.insert('', 'end', values=(
            proc['pid'], proc['user'], proc['cpu'], proc['mem'], proc['status'], proc['command']
        ), tags=tuple(tags))
```

# MEMORY MONITORING PART

1



2



1. Overall RAM usage will be showed here
2. This refers to the amount of Swap Space currently being used by the operating system.



## code

```
def update_memory_info(self):
    try:
        with open('/proc/meminfo', 'r') as f:
            lines = f.readlines()

        meminfo = {line.split(':')[0]: int(line.split(':')[1].strip().split()[0]) for line in lines}

        # RAM Calculation
        mem_total = meminfo.get('MemTotal', 1)
        mem_available = meminfo.get('MemAvailable', 1)
        mem_used = mem_total - mem_available
        mem_percent = (mem_used / mem_total) * 100

        self.mem_labels["RAM"]['bar']['value'] = mem_percent
        self.mem_labels["RAM"]['label']['text'] = f"Used: {mem_used/1024:.1f} MB / {mem_total/1024:.1f} MB ({mem_percent:.1f}%)"

        # Swap Calculation
        swap_total = meminfo.get('SwapTotal', 1)
        swap_free = meminfo.get('SwapFree', 1)
        swap_used = swap_total - swap_free
        swap_percent = 0
        if swap_total > 0:
            swap_percent = (swap_used / swap_total) * 100

        self.mem_labels["Swap"]['bar']['value'] = swap_percent
        self.mem_labels["Swap"]['label']['text'] = f"Used: {swap_used/1024:.1f} MB / {swap_total/1024:.1f} MB ({swap_percent:.1f}%)"

    except (IOError, ValueError):
        pass
```

# ACKNOWLEDGEMENT

The graphical user interface setup is built by using Ai tools like Gemini and Chat gpt



THANK YOU