

# System Monitor Tool

A tool that displays real-time information about system processes, memory usage, and CPU load ? similar to the 'top' command.

## Code

```
// sysmon.cpp

// Build: g++ -std=c++17 sysmon.cpp -o sysmon -lnurses
// Run: ./sysmon [interval_seconds]

#include <bits/stdc++.h>
#include <nurses.h>
#include <signal.h>
using namespace std;
namespace fs = filesystem;

struct ProcSnapshot {
    int pid;
    string name;
    unsigned long total_time = 0; // utime+stime (jiffies)
    long rss_kb = 0;
    double cpu_percent = 0.0;
    double mem_mb = 0.0;
};

unsigned long get_total_jiffies() {
    ifstream f("/proc/stat");
    string line;
    getline(f, line);
    // line: cpu 3357 0 4313 1362393 ...
    stringstream ss(line);
    string cpu; unsigned long v, sum=0;
    ss >> cpu;
    while (ss >> v) sum += v;
    return sum;
}

vector<int> list_pids() {
    vector<int> pids;
    for (auto &e : fs::directory_iterator("/proc")) {
        string name = e.path().filename();
        if (!name.empty() && all_of(name.begin(), name.end(), ::isdigit))
            pids.push_back(stoi(name));
    }
}
```

```

    return pids;
}

bool read_proc_times(int pid, unsigned long &totaltime, long &rss_kb, string &name) {
    string statp = "/proc/" + to_string(pid) + "/stat";
    ifstream sf(statp);
    if (!sf) return false;
    string content; getline(sf, content);
    size_t rp = content.rfind(' ');
    if (rp == string::npos) return false;
    name = content.substr(content.find(' ')+1, rp - content.find(' ') - 1);
    string after = content.substr(rp+2);
    vector<string> toks; string t; stringstream s(after);
    while (s >> t) toks.push_back(t);
    if (toks.size() < 13) return false;
    unsigned long utime = stoul(toks[11]);
    unsigned long stime = stoul(toks[12]);
    totaltime = utime + stime;

    // rss from /proc/[pid]/status
    rss_kb = 0;
    ifstream pf("/proc/" + to_string(pid) + "/status");
    string line;
    while (pf && getline(pf, line)) {
        if (line.rfind("VmRSS:", 0) == 0) {
            string key; long val; string unit;
            stringstream ss(line); ss >> key >> val >> unit;
            rss_kb = val;
            break;
        }
    }
    return true;
}

int main(int argc, char** argv) {
    int interval = 2;
    if (argc >= 2) interval = stoi(argv[1]);
    bool sort_by_cpu = true;

    initscr();
    cbreak();
    noecho();
    nodelay(stdscr, TRUE); // non-blocking getch
    keypad(stdscr, TRUE);

    // We'll keep previous snapshots in maps
    unordered_map<int, unsigned long> prev_proc_time;
    unsigned long prev_total_jiffies = get_total_jiffies();

```

```

while (true) {
    unsigned long cur_total_jiffies = get_total_jiffies();

    vector<ProcSnapshot> procs;

    for (int pid : list_pids()) {
        unsigned long ttime;
        long rss;
        string name;
        if (!read_proc_times(pid, ttime, rss, name)) continue;
        ProcSnapshot ps;
        ps.pid = pid; ps.name = name; ps.total_time = ttime; ps.rss_kb = rss; ps.mem_mb =
rss/1024.0;
        auto it = prev_proc_time.find(pid);
        unsigned long proc_prev = (it!=prev_proc_time.end()) ? it->second : 0;
        unsigned long proc_delta = (ttime >= proc_prev) ? (ttime - proc_prev) : 0;
        unsigned long total_delta = (cur_total_jiffies >= prev_total_jiffies) ?
(cur_total_jiffies - prev_total_jiffies) : 1;
        ps.cpu_percent = 100.0 * (double)proc_delta / (double)total_delta;
        procs.push_back(ps);
        // update temp map later
    }

    // update previous maps
    prev_proc_time.clear();
    for (auto &p : procs) prev_proc_time[p.pid] = p.total_time;
    prev_total_jiffies = cur_total_jiffies;

    if (sort_by_cpu) {
        sort(procs.begin(), procs.end(), [](auto &a, auto &b){ return a.cpu_percent >
b.cpu_percent; });
    } else {
        sort(procs.begin(), procs.end(), [](auto &a, auto &b){ return a.mem_mb >
b.mem_mb; });
    }

    // Render
    erase();
    mvprintw(0,0,"SysMon (Day 5) - refresh every %d s - sort by %s - press 's' to toggle
sort, 'k' to kill PID, 'q' to quit", interval, sort_by_cpu ? "CPU" : "MEM");
    mvprintw(1,0,"%-6s %-20s %-8s %-8s", "PID", "NAME", "CPU%", "MEM(MB)");
    int row = 2;
    int show = 0;
    for (auto &p : procs) {
        if (show++ >= 25) break;
        mvprintw(row++, 0, "%-6d %-20s %7.2f %-8.1f", p.pid, p.name.c_str(),
p.cpu_percent, p.mem_mb);
    }
}

```

```

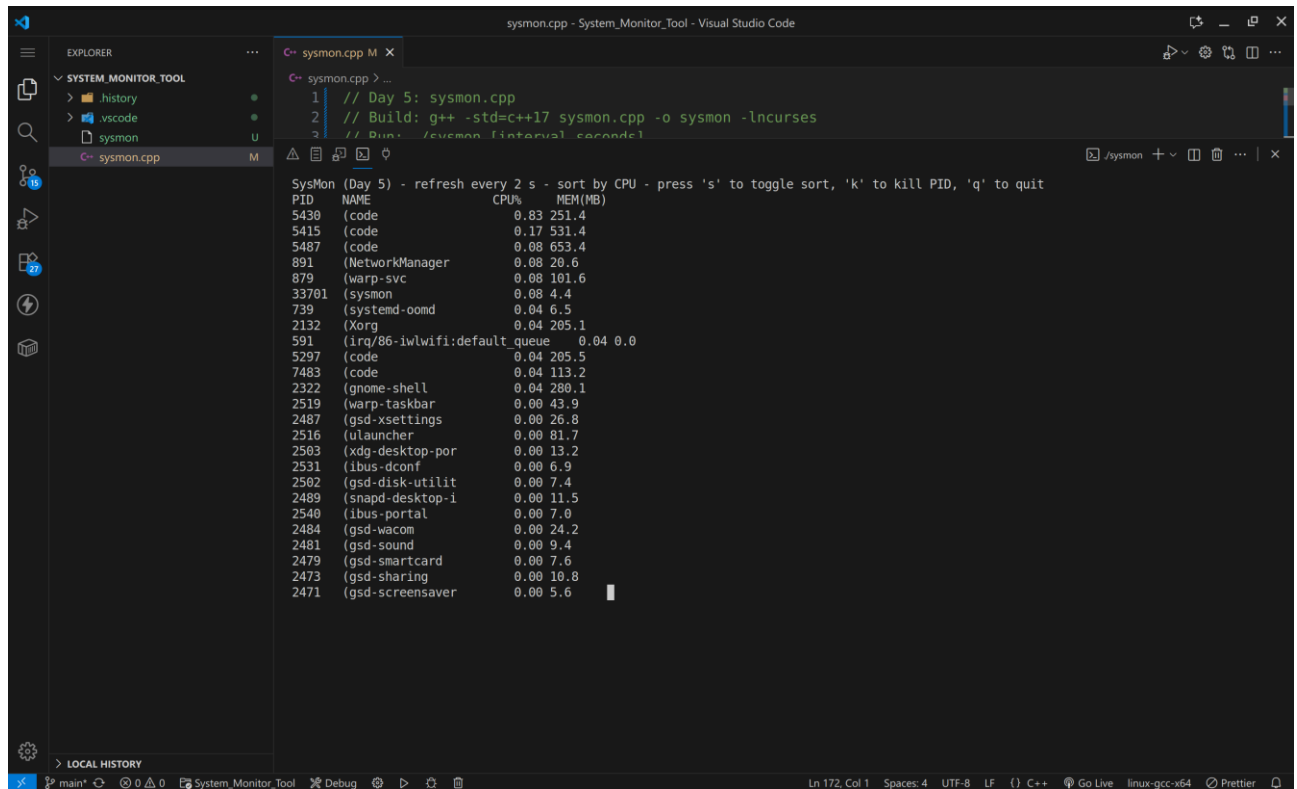
    }
    refresh();

    // handle keys
    int ch;
    bool killedSomething = false;
    for (int t=0; t<interval*10; ++t) { // poll every 100ms to be responsive to keypress
        ch = getch();
        if (ch == 'q') {
            endwin();
            return 0;
        } else if (ch == 's') {
            sort_by_cpu = !sort_by_cpu;
            break; // break to refresh immediately
        } else if (ch == 'k') {
            // prompt user for PID
            echo();
            nodelay(stdscr, FALSE);
            mvprintw(row+1, 0, "Enter PID to kill (SIGTERM): ");
            char buf[32];
            getnstr(buf, 31);
            int pid = atoi(buf);
            int res = kill(pid, SIGTERM);
            if (res == 0) mvprintw(row+2,0,"SIGTERM sent to %d", pid);
            else mvprintw(row+2,0,"Failed to kill %d: %s", pid, strerror(errno));
            noecho();
            nodelay(stdscr, TRUE);
            killedSomething = true;
            break;
        } else if (ch == ERR) {
            // no input
        }
        napms(100); // 100 ms
    }
    if (killedSomething) {
        // immediate refresh in next loop iteration
        continue;
    }
}

endwin();
return 0;
}

```

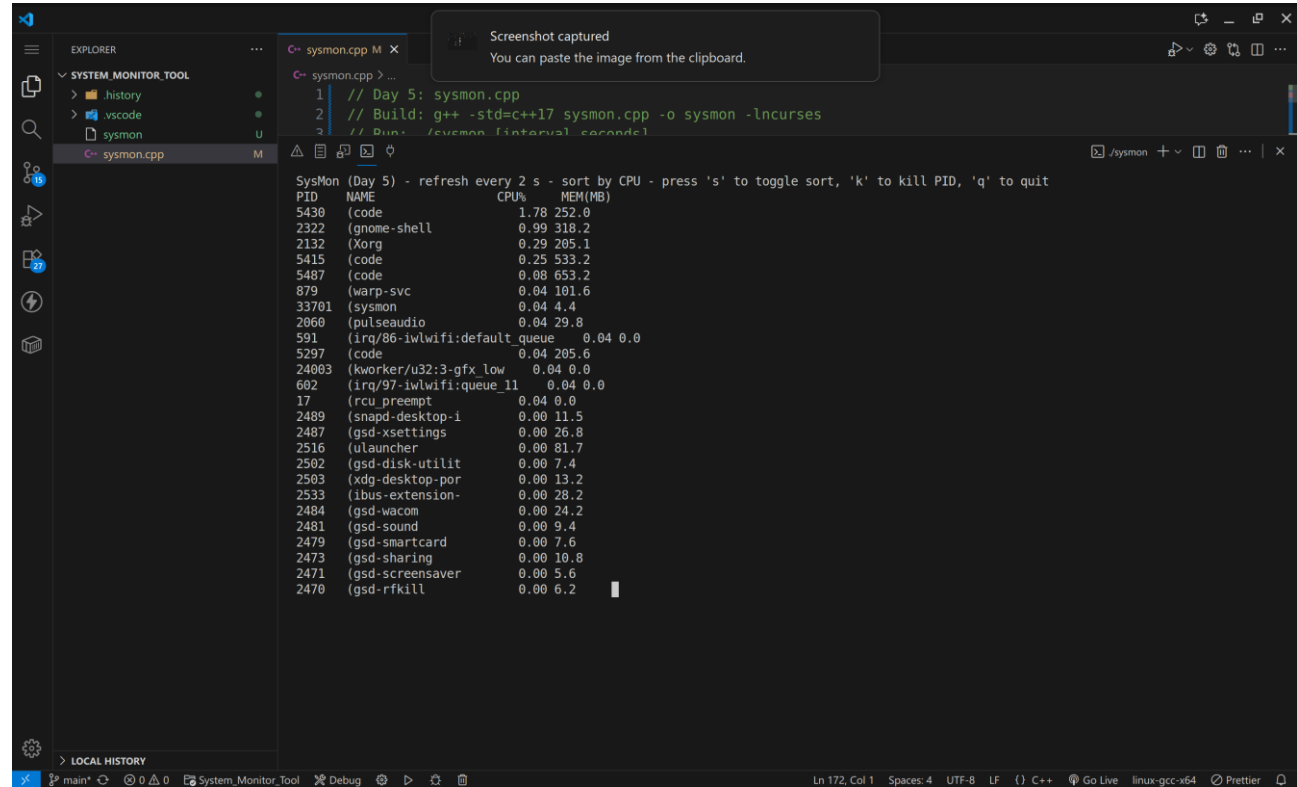
# Output



```
1 // Day 5: sysmon.cpp
2 // Build: g++ -std=c++17 sysmon.cpp -o sysmon -lnurses
3 // Run: ./sysmon [interval second]
```

SysMon (Day 5) - refresh every 2 s - sort by CPU - press 's' to toggle sort, 'k' to kill PID, 'q' to quit

PID	NAME	CPU%	MEM(MB)
5430	(code)	0.83	251.4
5415	(code)	0.17	531.4
5487	(code)	0.08	653.4
891	(NetworkManager)	0.08	20.6
879	(warp-svc)	0.08	101.6
33701	(sysmon)	0.08	4.4
739	(systemd-oomd)	0.04	6.5
2132	(Xorg)	0.04	205.1
591	(irq/86-iwlwifi:default_queue)	0.04	0.0
5297	(code)	0.04	205.5
7483	(code)	0.04	113.2
2322	(gnome-shell)	0.04	280.1
2519	(warp-taskbar)	0.00	43.9
2487	(gsd-xsettings)	0.00	26.8
2516	(ulauncher)	0.00	81.7
2503	(xdg-desktop-por)	0.00	13.2
2531	(ibus-dconf)	0.00	6.9
2502	(gsd-disk-utilit)	0.00	7.4
2489	(snapd-desktop-i)	0.00	11.5
2540	(ibus-portal)	0.00	7.0
2484	(gsd-wacom)	0.00	24.2
2481	(gsd-sound)	0.00	9.4
2479	(gsd-smartcard)	0.00	7.6
2473	(gsd-sharing)	0.00	10.8
2471	(gsd-screensaver)	0.00	5.6



```
1 // Day 5: sysmon.cpp
2 // Build: g++ -std=c++17 sysmon.cpp -o sysmon -lnurses
3 // Run: ./sysmon [interval second]
```

SysMon (Day 5) - refresh every 2 s - sort by CPU - press 's' to toggle sort, 'k' to kill PID, 'q' to quit

PID	NAME	CPU%	MEM(MB)
5430	(code)	1.78	252.0
2322	(gnome-shell)	0.99	318.2
2132	(Xorg)	0.29	205.1
5415	(code)	0.25	533.2
5487	(code)	0.08	653.2
879	(warp-svc)	0.04	101.6
33701	(sysmon)	0.04	4.4
2060	(pulseaudio)	0.04	29.8
591	(irq/86-iwlwifi:default_queue)	0.04	0.0
5297	(code)	0.04	205.6
24803	(kworker/u32:3-gfx_low)	0.04	0.0
602	(irq/97-iwlwifi:queue_11)	0.04	0.0
17	(rcu_preempt)	0.04	0.0
2489	(snapd-desktop-i)	0.00	11.5
2487	(gsd-xsettings)	0.00	26.8
2516	(ulauncher)	0.00	81.7
2502	(gsd-disk-utilit)	0.00	7.4
2503	(xdg-desktop-por)	0.00	13.2
2533	(ibus-extension-)	0.00	28.2
2484	(gsd-wacom)	0.00	24.2
2481	(gsd-sound)	0.00	9.4
2479	(gsd-smartcard)	0.00	7.6
2473	(gsd-sharing)	0.00	10.8
2471	(gsd-screensaver)	0.00	5.6
2470	(gsd-rfkill)	0.00	6.2

sysmon.cpp - System\_Monitor\_Tool - Visual Studio Code

```
1 // Day 5: sysmon.cpp
2 // Build: g++ -std=c++17 sysmon.cpp -o sysmon -lncurses
3 // Run: ./sysmon [interval seconds]
```

SysMon (Day 5) - refresh every 2 s - sort by CPU - press 's' to toggle sort, 'k' to kill PID, 'q' to quit

PID	NAME	CPU%	MEM(MB)
5430	(code)	1.62	252.0
34362	(idea)	1.21	1755.0
5415	(code)	1.16	533.4
2322	(gnome-shell)	0.33	279.8
2132	(Xorg)	0.29	205.0
5487	(code)	0.21	653.7
7501	(cpptools-srv)	0.17	261.0
879	(warp-svc)	0.12	101.9
34735	(tracker-extract)	0.12	30.7
7251	(cpptools)	0.04	60.0
7483	(code)	0.04	113.2
3629	(brave)	0.04	322.8
591	(irq/86-iwlwifi:default_queue)	0.04	0.0
3323	(brave)	0.04	139.8
2285	(gvfsd)	0.04	7.9
24003	(kworker/u32:3-gfx_low)	0.04	0.0
836	(accounts-daemon)	0.04	7.3
839	(avahi-daemon)	0.04	4.9
27863	(kworker/u32:5-events_unbound)	0.04	0.0
2080	(dbus-daemon)	0.04	5.2
2920	(gjs)	0.04	62.0
891	(NetworkManager)	0.04	20.6
34513	(fsnotifier)	0.04	0.0
17	(rcu_preempt)	0.04	0.0
34734	(sysmon)	0.04	4.4

Enter PID to kill (SIGTERM): 34362

sysmon.cpp - System\_Monitor\_Tool - Visual Studio Code

```
1 // Day 5: sysmon.cpp
2 // Build: g++ -std=c++17 sysmon.cpp -o sysmon -lncurses
3 // Run: ./sysmon [interval seconds]
```

SysMon (Day 5) - refresh every 2 s - sort by CPU - press 's' to toggle sort, 'k' to kill PID, 'q' to quit

PID	NAME	CPU%	MEM(MB)
5430	(code)	2.86	251.7
2132	(Xorg)	0.62	205.0
2322	(gnome-shell)	0.50	336.9
34982	(tracker-extract)	0.33	31.6
94	(kcompactd0)	0.33	0.0
5460	(code)	0.25	166.9
115	(kswapd0)	0.25	0.0
5415	(code)	0.21	535.2
5487	(code)	0.08	653.5
891	(NetworkManager)	0.08	20.6
2051	(systemd)	0.08	9.9
2823	(tracker-miner-f)	0.08	41.6
34734	(sysmon)	0.08	4.4
2920	(gjs)	0.04	62.0
27863	(kworker/u32:5-events_unbound)	0.04	0.0
7483	(code)	0.04	113.7
5297	(code)	0.04	209.6
2080	(dbus-daemon)	0.04	5.2
2502	(gsd-disk-utilit)	0.00	7.4
2473	(gsd-sharing)	0.00	10.8
2489	(snapd-desktop-i)	0.00	11.5
2487	(gsd-xsettings)	0.00	26.8
2484	(gsd-wacom)	0.00	24.2
2481	(gsd-sound)	0.00	9.4
2479	(gsd-smartcard)	0.00	7.6

## **GitHub**

**Repo URL:**

**[https://github.com/SauvikPrasad/Wipro\\_System\\_Monitor\\_Tool](https://github.com/SauvikPrasad/Wipro_System_Monitor_Tool)**

**Sanjay Bhakat**

**Regd. No. : 2241002141**

**Wipro COE Embedded Batch 1**