PROPOSAL

MONITORING RESOURCE SYSTEM

(LINUX TASK MANAGER ADVANCED)

Mata Kuliah Sistem Operasi

Dosen Pengampu:

Ferdi Cahyadi, S.Kom., M.Cs

Anggota Kelompok:

Riki Andika Saputra	2401020134
Muhamad Isra Dwi Firmansya	2401020143
Devina Rindumasha	2401020153
Bunga Salsabila Pebriyani	2401020150

PRODI TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN
UNIVERSITAS MARITIM RAJA ALI HAJI

2025

BAB I
PENDAHULUAN

1.1 Latar Belakang

Era komputasi modern menuntut sistem operasi untuk memiliki kemampuan dalam mengelola sumber daya secara efisien. Pada komputer, performa dapat menurun akibat penggunaan CPU yang terlalu tinggi, kebocoran memori (memory leak), proses yang tidak terkendali, atau konsumsi disk yang melebihi kapasitas. Tanpa adanya mekanisme monitoring yang baik, masalah tersebut dapat menyebabkan *lag*, penurunan kinerja, hingga kegagalan sistem (*system crash*).

Linux sebagai salah satu sistem operasi yang paling banyak digunakan, terutama pada server dan sistem berbasis jaringan, menyediakan berbagai fitur untuk memantau aktivitas sistem melalui antarmuka seperti `/proc`, *system calls*, dan sinyal proses. Namun, tools bawaan seperti `top` atau `htop` terkadang terlalu kompleks untuk analisis tertentu atau kurang fleksibel untuk pembelajaran akademik. Hal ini membuka peluang bagi mahasiswa untuk membangun aplikasi monitoring yang dapat disesuaikan dengan kebutuhan belajar.

Proyek *Monitoring Resource System* ini bertujuan untuk mengembangkan aplikasi yang mampu menampilkan penggunaan CPU, RAM, disk, serta daftar proses secara real-time. Sistem juga dilengkapi kemampuan menghentikan proses menggunakan *signal handling* (`SIGTERM`, `SIGKILL`), sesuai konsep manajemen proses di mata kuliah Sistem Operasi. Dengan membangun aplikasi ini, mahasiswa memperoleh pemahaman praktis mengenai penjadwalan proses, mekanisme interupsi, serta struktur data dan antarmuka kernel dalam mengontrol proses.

Selain itu, proyek ini memberikan pengalaman nyata mulai dari tahap perencanaan, implementasi teknis, hingga penyusunan laporan akhir. Diharapkan aplikasi ini bermanfaat tidak hanya sebagai tugas akademik, tetapi juga sebagai dasar untuk memahami pengembangan sistem monitoring profesional.

1.2 Rumusan Masalah

1. Bagaimana merancang aplikasi monitoring yang dapat menampilkan penggunaan CPU, RAM, dan disk secara real-time?
2. Bagaimana menampilkan daftar proses aktif lengkap dengan PID dan resource usage?
3. Bagaimana mengimplementasikan fitur *kill process* menggunakan *signal handling* Linux?
4. Bagaimana memastikan aplikasi berjalan stabil saat dilakukan *stress test*?

1.3 Tujuan

1. Mengembangkan aplikasi monitoring sederhana berbasis Linux.
2. Menampilkan data penggunaan CPU, RAM, dan disk secara real-time.
3. Menyediakan fitur penghentian proses menggunakan sinyal (`kill`, `SIGTERM`, `SIGKILL`).
4. Melatih mahasiswa memahami konsep manajemen proses dan resource pada sistem operasi.
5. Menghasilkan laporan proyek sesuai standar akademik.

BAB II
DASAR TEORI

2.1 Sistem Operasi Linux

Linux adalah sistem operasi berbasis kernel monolitik yang dirancang untuk memberikan mekanisme multitasking, multiuser, dan pengelolaan resource secara efisien. Linux menyediakan filesystem khusus bernama /proc, yaitu virtual filesystem yang digunakan untuk menyimpan informasi mengenai proses, memori, CPU, disk, serta berbagai parameter kernel. Data di dalam /proc diperbarui secara real-time sehingga sangat cocok digunakan untuk aplikasi monitoring resource. Selain itu, Linux juga memiliki sistem scheduling yang mengatur proses mana yang berjalan pada CPU menggunakan algoritma seperti Completely Fair Scheduler (CFS), yang mempertahankan fairness antar proses.

2.2 Manajemen Proses dalam Linux

Semua aktivitas yang berjalan di Linux dianggap sebagai **proses**. Setiap proses memiliki identitas unik yang disebut **PID (Process ID)** dan dikendalikan oleh struktur internal bernama **Process Control Block (PCB)**. PCB menyimpan informasi penting seperti state proses, register CPU, prioritas, penggunaan memori, dan pointer ke parent/child process.
Proses pada Linux memiliki beberapa state utama:

1. **Running** – proses sedang dieksekusi CPU.
2. **Ready** – proses siap dieksekusi tetapi sedang menunggu giliran.
3. **Blocked/Waiting** – proses menunggu I/O atau event lain.
4. **Terminated** – proses selesai.
5. **Zombie** – proses selesai tetapi masih memiliki entry PCB.

Informasi mengenai seluruh proses dapat dilihat melalui command seperti **ps**, **top**, atau **htop**, namun pada proyek ini, program monitoring akan membaca data tersebut langsung dari sistem.

2.3 Monitoring Resource pada Linux

Linux menyediakan berbagai API dan file virtual yang memungkinkan pengambilan data resource secara akurat. Beberapa bagian penting yang digunakan antara lain:

a. **CPU Usage**
   Informasi CPU diperoleh dari file /proc/stat, khususnya baris yang diawali cpu. Data ini memuat waktu idle, user, system, dan iowait yang selanjutnya dihitung untuk mendapatkan persentase penggunaan CPU.

b. **Memori (RAM) Usage**
   Informasi RAM tersedia pada /proc/meminfo, yang memberikan data seperti MemTotal, MemFree, dan Buffers. Perhitungan dilakukan untuk mendapatkan persentase pemakaian memori.
c. **Disk Usage dan I/O**
   Data disk diambil dari /proc/diskstats atau melalui fungsi library. Informasi ini mencakup jumlah baca/tulis, total request, dan waktu akses disk.
d. **Daftar Proses**
   Setiap proses di Linux memiliki folder /proc/PID/ yang berisi detail proses, termasuk nama executable, penggunaan CPU, penggunaan memori, dan status proses. Aplikasi monitoring akan melakukan pembacaan berkala terhadap folder ini untuk menampilkan proses aktif.

2.4 Signal Handling pada Linux

Signal adalah mekanisme komunikasi antar-proses pada sistem operasi. Sinyal digunakan untuk memberi perintah atau notifikasi kepada proses, misalnya untuk menghentikan, mem-pause, atau memaksa proses keluar. Beberapa sinyal penting yang digunakan dalam proyek ini antara lain:

a. **SIGTERM (15)** – menghentikan proses secara normal.
b. **SIGKILL (9)** – memaksa proses berhenti tanpa bisa ditolak.
c. **SIGSTOP (19)** – menghentikan proses sementara.
d. **SIGCONT (18)** – melanjutkan proses yang di-pause.

Pada bahasa C, sinyal dikirim menggunakan fungsi **kill(pid, signal)**. Pada Python, sinyal dikirim menggunakan **os.kill(pid, signal)**. Tidak semua proses dapat dihentikan oleh user biasa; beberapa proses membutuhkan *root privilege*. Hal ini penting untuk dijelaskan pada laporan sebagai bagian dari permasalahan hak akses.

2.5 Library dan Tools Pendukung

Dalam implementasi monitoring ini, terdapat dua pendekatan umum:

1. **Python + psutil**
   Library psutil menyediakan API sederhana untuk memantau CPU, memori, disk, dan proses tanpa perlu membaca file /proc secara manual. Psutil banyak digunakan untuk aplikasi monitoring tingkat tinggi karena stabil, cepat, dan mudah digunakan.
2. **C + Akses langsung ke /proc**
   Pendekatan ini menawarkan performa lebih tinggi dan kontrol penuh terhadap sistem. Dengan membaca file seperti /proc/stat, /proc/meminfo, dan /proc/[PID]/stat, programmer dapat membuat monitor yang lebih efisien seperti “top” versi custom.

Tools ini digunakan untuk mengimplementasikan fitur utama proyek, yaitu pemantauan resource real-time, penampilan data proses, serta kemampuan untuk menghentikan proses melalui signal handling.

BAB III

METODOLOGI, PERENCANAAN TEKNIS DAN IMPLEMENTASI

3.1 Arsitektur Sistem

Arsitektur sistem yang diusulkan terdiri dari beberapa komponen utama yang saling terintegrasi untuk melakukan monitoring resource secara real-time. Adapun komponen-komponen tersebut adalah sebagai berikut:

a. **Data Collector (Polling Engine)**
   Bertugas mengambil snapshot penggunaan CPU, RAM, disk, dan proses setiap interval waktu tertentu (default 1 detik). Data diperoleh melalui pembacaan terhadap *kernel interface* seperti /proc atau menggunakan pustaka pendukung.
b. **Data Store (Ring Buffer Storage)**
   Digunakan untuk menyimpan sejumlah data terbaru (N sampel) sebagai basis penampilan grafik dan riwayat. Penyimpanan bersifat *circular* sehingga efisien dalam penggunaan memori.
c. **UI/Display Module**
   Menyajikan informasi dalam bentuk tabel dan grafik sederhana melalui antarmuka CLI. Dapat menggunakan *library* seperti curses atau rich. Opsi pengembangan GUI atau web dashboard bersifat tambahan.
d. **Process Controller**
   Modul yang bertanggung jawab dalam validasi PID dan pengiriman sinyal untuk menghentikan proses. Fitur ini mendukung pengiriman sinyal aman (SIGTERM) maupun sinyal paksa (SIGKILL).
e. **Logger / Export Engine**
   Menyimpan riwayat monitoring ke format CSV atau JSON. Digunakan untuk kebutuhan analisis setelah pengujian atau demo berlangsung.

3.2 Spesifikasi Fungsional Utama

Sistem monitoring ini memiliki beberapa fungsi inti sebagai berikut:

a. Menampilkan penggunaan CPU per inti dan total dalam persentase.
b. Menampilkan status penggunaan RAM meliputi total, used, free, dan persentase pemakaian.
c. Memberikan informasi penggunaan disk serta ringkasan aktivitas I/O.
d. Menampilkan daftar proses lengkap berisi PID, user, nama proses, CPU%, dan penggunaan memori.
e. Memberikan opsi untuk menghentikan proses melalui pengiriman sinyal SIGTERM atau SIGKILL dengan mekanisme konfirmasi.
f. Menyimpan historis penggunaan resource dan menyediakan fitur export ke CSV.
g. Menampilkan grafik sederhana untuk CPU dan RAM berdasarkan data ring buffer.

3.3 Spesifikasi Non-Fungsional

Beberapa spesifikasi non-fungsional yang menjadi batasan sistem adalah:

a. **Interval Pengambilan Data:** Default 1 detik, dapat disesuaikan oleh pengguna.
b. **Efisiensi Sistem:** Aplikasi monitor harus menggunakan kurang dari 5% CPU pada kondisi normal agar tidak membebani sistem.
c. **Kompatibilitas:** Sistem kompatibel dengan kernel Linux modern (Debian/Ubuntu) serta mendukung eksekusi pada environment Linux native maupun WSL.

3.4 Rencana Implementasi Mingguan

Minggu 11

- Penyusunan proposal dan perancangan arsitektur sistem secara menyeluruh.

Minggu 12

- Instalasi Linux / WSL / VM.
- Persiapan environment coding menggunakan Python atau C.
- Pembuatan *skeleton project* dan penataan struktur repository.

Minggu 13

- Implementasi Data Collector untuk CPU dan RAM.
- Menampilkan hasil monitoring dasar pada antarmuka CLI.
- Penambahan modul penyimpanan ring buffer.

Minggu 14

- Implementasi Process Viewer dan Process Controller.
- Pengujian fitur penghentian proses (kill) menggunakan SIGTERM dan SIGKILL.
- Melakukan stress test menggunakan aplikasi beban (misal: stress-ng) dan melakukan tuning error handling.

Minggu 15

- Pembuatan grafik CPU/RAM (export file PNG/CSV).
- Penyempurnaan logging dan output akhir.
- Penyusunan laporan dan persiapan demo program.

3.5 Pengujian & Evaluasi

Pengujian yang dilakukan terdiri dari:

a. **Unit Test:**
   Menguji keakuratan pembacaan nilai CPU, RAM, dan proses.
b. **Stress Test:**
   Menguji kestabilan sistem ketika beban CPU atau memori tinggi, termasuk pengujian fitur kill process.
c. **Evaluasi Overhead:**
   Mengukur seberapa besar penggunaan CPU% yang dihasilkan oleh program monitor sendiri.

3.6 Manajemen Risiko & Mitigasi

a. **Hak Akses Proses:**
   Menghentikan proses milik user lain memerlukan hak akses lebih tinggi.
   Mitigasi: Batasi aksi hanya pada proses milik user atau jalankan aplikasi dengan sudo di lingkungan uji.
b. **Overhead Pengumpulan Data:**
   Interval polling terlalu kecil dapat menambah beban CPU.
   Mitigasi: Ubah interval sesuai performa dan kapasitas perangkat.
c. **Data Tidak Konsisten:**
   Pembacaan /proc dapat gagal pada proses yang baru saja mati.
   Mitigasi: Tambahkan validasi dan fallback ke pustaka psutil.
d. **Kesalahan Penghentian Proses:**
   Risiko menghentikan proses penting sistem.
   Mitigasi: Tambahkan konfirmasi serta daftar PID yang tidak boleh dihentikan.

3.7 Sumber Daya & Anggaran

a. **Perangkat keras:** Laptop/PC mahasiswa (sudah tersedia).
b. **Perangkat lunak:** Linux, Python/C, psutil (gratis / open-source).
c. **Anggaran tambahan:** Biaya cetak dan jilid laporan (opsional).

3.8 Output Akhir

a. Program monitoring berjalan (CLI atau tambahan GUI).
b. Fitur manajemen proses (kill SIGTERM/SIGKILL).
c. Grafik penggunaan resource dan export data.
d. Laporan lengkap beserta demo program.

KESIMPULAN

Dari keseluruhan proses perencanaan sampai tahap implementasi, project ***Monitoring Resource System*** ini menunjukkan bahwa Linux memiliki fasilitas yang sangat lengkap untuk memantau aktivitas sistem, khususnya melalui direktori /proc dan mekanisme sinyal proses. Aplikasi yang dibuat mampu menampilkan penggunaan CPU, RAM, disk, hingga daftar proses secara langsung(real time), sekaligus menyiapkan fitur untuk menghentikan proses menggunakan sinyal **SIGTERM** maupun **SIGKILL**. Dengan kata lain, sistem ini tidak hanya berfungsi sebagai alat pemantauan, tetapi juga bisa membantu dalam pengendalian proses ketika diperlukan.

DAFTAR PUSTAKA

Love, R. (2010). *Linux Kernel Development.* Indianapolis: Addison-Wesley.

Robbins, K. A., & Robbins, S. (2003). *UNIX Systems Programming: Communication, Concurrency, and Threads.* Upper Saddle River: Prentice Hall.

Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). *Operating System Concepts.* Hoboken: Wiley.

Stallings, W. (2018). *Operating Systems: Internals and Design Principles.* Boston: Pearson.

Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems.* Upper Saddle River: Pearson.