LAPORAN AKHIR

IMPLEMENTASI KILL PROCESS DAN PENGUJIAN STRESS

Dosen Pengampu:

Ferdi Chahyadi, S.Kom., M.Cs

Disusun Oleh:

Siti Umayah	2401020018
Anika	2401020029
Aldi Syaputra	2401020024
Fauziah Sal Sabillah	2401020034

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN

UNIVERSITAS MARITIM RAJA ALI HAJI

TANJUNGPINANG

2025

ABSTARK

Laporan ini merupakan dokumentasi lengkap pelaksanaan proyek Enhanced Resource Monitoring System yang dilaksanakan selama 3 minggu (minggu 12-14). Proyek ini mengembangkan aplikasi monitoring resource berbasis Linux yang mampu memantau CPU, RAM, disk, dan proses secara real-time dengan visualisasi yang informatif.

Minggu 12 fokus pada instalasi Linux Ubuntu 22.04 LTS dan setup environment development termasuk instalasi Python 3.10 dan library psutil 5.9.5. Testing dasar menunjukkan environment siap untuk development dengan semua dependencies terinstall dengan baik.

Minggu 13 merupakan fase implementasi utama dimana semua modul monitoring dikembangkan. Arsitektur modular diterapkan dengan lima file Python: monitor.py (orchestrator), cpu_monitor.py, ram_monitor.py, disk_monitor.py, dan process_monitor.py. Fitur yang diimplementasikan mencakup CPU monitoring per-core, RAM dengan buffers/cache, disk usage untuk multiple partisi, dan top 5 processes dengan grafik ASCII horizontal. Visualisasi menggunakan progress bar dan bar chart untuk kemudahan interpretasi data.

Minggu 14 fokus pada testing komprehensif dan optimization. Testing fungsional menunjukkan 100% pass rate (12/12 test cases). Performance testing menunjukkan overhead minimal dengan CPU <2% dan memory footprint ~25MB. Stress testing dengan CPU load (4 workers) dan memory load (1GB allocation) membuktikan aplikasi stabil dalam kondisi high load. Validasi akurasi dengan cross-comparison terhadap tools native (top, htop, free) menunjukkan akurasi >99% untuk semua metrics.

Hasil akhir adalah aplikasi monitoring yang lightweight, accurate, comprehensive, dan robust yang dapat digunakan untuk system administration dan troubleshooting performance issues. Aplikasi berhasil memenuhi semua requirements dengan excellent performance.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang semakin pesat menuntut sistem komputer untuk selalu berada dalam kondisi optimal. Sistem operasi memiliki peranan yang sangat penting sebagai pengelola utama sumber daya komputer, seperti Central Processing Unit (CPU), Random Access Memory (RAM), media penyimpanan (disk), serta proses yang berjalan di dalam sistem. Pengelolaan sumber daya yang tidak optimal dapat menyebabkan penurunan kinerja sistem, terjadinya bottleneck, bahkan kegagalan sistem dalam menjalankan aplikasi.

Pada sistem operasi Linux, informasi mengenai penggunaan sumber daya sebenarnya dapat diakses melalui berbagai mekanisme, seperti filesystem virtual /proc dan /sys, serta melalui perintah bawaan seperti top, htop, dan free. Namun, penggunaan mekanisme tersebut masih memerlukan pemahaman teknis yang cukup mendalam dan kurang praktis bagi pengguna pemula. Oleh karena itu, diperlukan sebuah aplikasi monitoring yang mampu menyajikan informasi sumber daya sistem secara ringkas, jelas, dan mudah dipahami.

Berdasarkan kebutuhan tersebut, proyek ini mengembangkan sebuah aplikasi *Monitoring Resource System* berbasis Linux. Aplikasi ini dirancang untuk memantau penggunaan CPU, RAM, disk, serta proses yang berjalan secara *real-time*, disertai dengan visualisasi berupa *progress bar* dan grafik ASCII yang informatif. Aplikasi dibangun menggunakan bahasa pemrograman Python dengan memanfaatkan library psutil, yang menyediakan abstraksi tingkat tinggi untuk mengakses informasi sistem secara efisien dan akurat.

Pelaksanaan proyek ini dilakukan selama tiga minggu, yaitu pada minggu ke-12 hingga minggu ke-14 perkuliahan. Minggu ke-12 difokuskan pada instalasi sistem operasi Linux dan persiapan lingkungan pengembangan. Minggu ke-13 berfokus pada implementasi modul-modul monitoring, sedangkan minggu ke-14 difokuskan pada pengujian, *stress testing*, serta optimasi aplikasi. Diharapkan melalui proyek ini, mahasiswa dapat memahami secara praktis konsep manajemen sumber daya pada sistem operasi Linux sekaligus meningkatkan keterampilan pengembangan aplikasi sistem.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam proyek ini adalah sebagai berikut:

1. Bagaimana melakukan instalasi sistem operasi Linux dan menyiapkan *development environment* yang optimal untuk pengembangan aplikasi monitoring?
2. Bagaimana merancang arsitektur aplikasi yang bersifat modular, terstruktur, dan mudah untuk dikembangkan lebih lanjut?
3. Bagaimana mengimplementasikan pemantauan CPU, RAM, disk, dan proses secara *real-time* menggunakan bahasa pemrograman Python?

4. Bagaimana menyajikan data hasil monitoring dalam bentuk visualisasi yang informatif dan mudah dipahami oleh pengguna?
5. Bagaimana melakukan pengujian aplikasi untuk memastikan kestabilan, keakuratan, dan performa sistem?
6. Bagaimana memvalidasi hasil monitoring dengan membandingkannya terhadap tools bawaan Linux?

1.3 Tujuan Proyek

1.3.1 Tujuan Umum

Tujuan umum dari proyek ini adalah mengembangkan sebuah aplikasi *Monitoring Resource System* berbasis Linux yang mampu memantau penggunaan CPU, RAM, disk, dan proses secara *real-time* dengan tampilan visual yang informatif.

1.3.2 Tujuan Khusus

Adapun tujuan khusus dari proyek ini adalah sebagai berikut:

1. Melakukan instalasi sistem operasi Linux dan menyiapkan lingkungan Python beserta library pendukung, khususnya psutil.
2. Merancang arsitektur aplikasi yang modular dengan menerapkan prinsip *separation of concerns*.
3. Mengimplementasikan modul monitoring CPU, baik secara keseluruhan maupun per-core.
4. Mengimplementasikan modul monitoring RAM yang mencakup penggunaan memori, buffer, cache, dan swap.
5. Mengimplementasikan modul monitoring disk untuk beberapa partisi penyimpanan.
6. Mengimplementasikan monitoring proses dengan menampilkan proses teratas berdasarkan penggunaan sumber daya.
7. Melakukan pengujian fungsional dan *stress testing* untuk memastikan aplikasi berjalan stabil.
8. Memvalidasi keakuratan data monitoring dengan membandingkan hasil aplikasi terhadap tools native Linux.

1.4 Manfaat Proyek

1.4.1 Manfaat Akademis

1. Memberikan pemahaman praktis mengenai konsep sistem operasi Linux, khususnya manajemen sumber daya dan proses.
2. Meningkatkan kemampuan mahasiswa dalam pengembangan aplikasi sistem menggunakan bahasa pemrograman Python.

3. Melatih penerapan konsep pemrograman modular dan perancangan arsitektur perangkat lunak.
4. Meningkatkan keterampilan dalam melakukan pengujian, debugging, dan optimasi aplikasi.

1.4.2 Manfaat Praktis

1. Menghasilkan sebuah aplikasi yang dapat digunakan sebagai alat bantu monitoring sistem pada lingkungan Linux.
2. Menjadi dasar pengembangan aplikasi monitoring yang lebih kompleks di masa mendatang.
3. Menjadi portofolio proyek yang dapat mendukung pengembangan karier di bidang teknologi informasi.

BAB II

PERANCANGAN ARSITEKTUR

2.1 Gambaran Umum Arsitektur Sistem

Pada bab ini dibahas mengenai perancangan arsitektur dari aplikasi *Monitoring Resource System*. Arsitektur sistem dirancang dengan pendekatan modular agar setiap bagian aplikasi memiliki fungsi dan tanggung jawab yang jelas. Pendekatan ini bertujuan untuk memudahkan proses pengembangan, pengujian, pemeliharaan, serta pengembangan lanjutan di masa depan.

Aplikasi monitoring ini berjalan pada sistem operasi Linux dan dikembangkan menggunakan bahasa pemrograman Python. Sistem bekerja dengan cara mengambil data penggunaan sumber daya komputer secara *real-time* melalui library psutil, kemudian mengolah dan menampilkannya ke dalam bentuk visualisasi teks (ASCII) pada terminal. Seluruh proses berjalan secara periodik sesuai dengan interval waktu yang telah ditentukan.

Secara umum, arsitektur sistem terdiri dari satu modul utama sebagai pengendali alur program (*orchestrator*) dan beberapa modul pendukung yang masing-masing bertanggung jawab terhadap pemantauan satu jenis sumber daya sistem.

2.2 Desain Arsitektur Modular

2.2.1 Konsep Modularisasi

Modularisasi merupakan teknik pemecahan sistem ke dalam beberapa modul kecil yang saling terpisah namun tetap terintegrasi. Dalam aplikasi ini, setiap modul dirancang untuk menangani satu aspek monitoring sumber daya, sehingga perubahan pada satu modul tidak akan memengaruhi modul lainnya secara langsung.

Pendekatan modular ini memberikan beberapa keuntungan, antara lain:

1. Kemudahan Pemeliharaan

   Kesalahan atau bug dapat dilacak dan diperbaiki pada modul tertentu tanpa memengaruhi keseluruhan sistem.

2. Kemudahan Pengembangan

   Fitur baru dapat ditambahkan dengan membuat modul baru tanpa mengubah struktur utama aplikasi.

3. Keterbacaan Kode

   Struktur kode menjadi lebih rapi dan mudah dipahami.

2.2.2 Prinsip Perancangan Sistem

Dalam perancangan aplikasi *Monitoring Resource System*, diterapkan beberapa prinsip dasar rekayasa perangkat lunak, yaitu:

1. Separation of Concerns
   Setiap modul hanya menangani satu jenis tugas spesifik, seperti monitoring CPU,

RAM, disk, atau proses. Hal ini bertujuan untuk mengurangi ketergantungan antar modul.

2. Single Responsibility Principle (SRP)
   Setiap file Python memiliki satu tanggung jawab utama. Sebagai contoh, modul cpu_monitor.py hanya bertanggung jawab untuk menampilkan informasi penggunaan CPU.

3. Don't Repeat Yourself (DRY)
   Fungsi-fungsi yang bersifat umum dan sering digunakan, seperti pembersihan layar, pembuatan *progress bar*, dan format ukuran memori, diletakkan pada modul utils.py agar tidak terjadi duplikasi kode.

4. Error Handling
   Setiap modul dilengkapi dengan mekanisme penanganan kesalahan (*try-except*) untuk mencegah aplikasi berhenti secara tiba-tiba akibat error sistem atau keterbatasan izin akses.

2.3 Struktur Modul Aplikasi

Aplikasi *Monitoring Resource System* terdiri dari beberapa file Python dengan fungsi sebagai berikut:

1. monitor.py
   Merupakan file utama yang berperan sebagai pengendali alur program. Modul ini bertugas memanggil seluruh modul monitoring, mengatur interval pembaruan data, serta menampilkan hasil monitoring secara terstruktur di terminal.

2. cpu_monitor.py
   Modul ini bertanggung jawab untuk mengambil dan menampilkan informasi penggunaan CPU, baik secara keseluruhan maupun per-core. Data ditampilkan dalam bentuk persentase dan *progress bar*.

3. ram_monitor.py
   Modul ini digunakan untuk memantau penggunaan memori sistem, termasuk total RAM, RAM terpakai, buffer, cache, dan swap. Informasi disajikan dalam satuan yang mudah dipahami (KB, MB, GB).

4. disk_monitor.py
   Modul ini berfungsi untuk menampilkan informasi penggunaan disk pada setiap partisi yang terdeteksi, termasuk total kapasitas, ruang terpakai, dan ruang kosong.

5. process_monitor.py
   Modul ini menampilkan daftar proses yang sedang berjalan dengan penggunaan sumber daya tertinggi, seperti CPU dan memori, sehingga membantu dalam proses analisis performa sistem.

6. utils.py
   Modul utilitas yang berisi fungsi-fungsi umum seperti pembersihan layar terminal, pembuatan *progress bar*, pengambilan timestamp, serta format ukuran memori.

2.4 Hubungan Antar Modul

Hubungan antar modul dalam sistem ini bersifat terpusat pada file monitor.py. Modul utama ini melakukan pemanggilan terhadap modul-modul monitoring lainnya tanpa adanya ketergantungan langsung antar modul monitoring.

Contoh hubungan antar modul adalah sebagai berikut:

1. monitor.py memanggil fungsi dari cpu_monitor.py, ram_monitor.py, disk_monitor.py, dan process_monitor.py.
2. Modul cpu_monitor.py, ram_monitor.py, dan disk_monitor.py memanfaatkan fungsi bantuan dari utils.py.
3. Modul process_monitor.py berdiri sendiri tanpa ketergantungan langsung pada modul lain.

   Dengan struktur seperti ini, arsitektur aplikasi menjadi lebih fleksibel, terorganisir, dan mudah untuk dikembangkan lebih lanjut.

2.5 Alur Kerja Sistem

Alur kerja aplikasi *Monitoring Resource System* secara umum adalah sebagai berikut:

1. Program dijalankan melalui file monitor.py.
2. Sistem membersihkan layar terminal dan menampilkan waktu (*timestamp*) saat ini.
3. Modul monitoring CPU, RAM, disk, dan proses dipanggil secara berurutan.
4. Data penggunaan sumber daya diambil menggunakan library psutil.
5. Data diolah dan ditampilkan dalam bentuk visualisasi teks pada terminal.
6. Program menunggu sesuai interval waktu yang telah ditentukan, kemudian mengulangi proses monitoring.

Dengan alur kerja tersebut, aplikasi mampu menampilkan kondisi sistem secara *real-time* dan berkelanjutan.

BAB III

IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini membahas tahapan implementasi *Monitoring Resource System* yang dilaksanakan selama tiga minggu, yaitu minggu ke-12 hingga minggu ke-14. Pembahasan meliputi proses instalasi sistem, pengembangan modul monitoring, pengujian fungsional, *stress testing*, serta optimasi sistem.

3.1 Implementasi Minggu ke-12: Instalasi Sistem dan Persiapan Lingkungan

3.1.1 Spesifikasi Perangkat Keras dan Lunak

Implementasi sistem dilakukan pada perangkat dengan spesifikasi sebagai berikut:

- **Processor** : AMD Athlon 300U with Radeon Vega Mobile Gfx (2.40 GHz)
- **RAM** : 8 GB
- **Media Penyimpanan** : 512 GB SSD
- **Sistem Operasi** : Ubuntu 22.04 LTS
- **Bahasa Pemrograman** : Python 3.10

Spesifikasi tersebut dinilai mencukupi untuk menjalankan sistem operasi Linux serta aplikasi monitoring yang dikembangkan.

3.1.2 Instalasi Sistem Operasi Linux

Tahapan instalasi sistem operasi Linux dilakukan sebagai berikut:

1. Mengunduh file ISO Ubuntu 22.04 LTS dari situs resmi.
2. Membuat media instalasi menggunakan *bootable USB* dengan aplikasi Rufus.
3. Melakukan proses *booting* dari USB dan menjalankan instalasi Ubuntu.
4. Melakukan konfigurasi partisi, zona waktu, dan akun pengguna.
5. Menyelesaikan proses instalasi hingga sistem dapat digunakan secara normal.
6. Melakukan pembaruan sistem menggunakan perintah:

   sudo apt update && sudo apt upgrade

Hasil dari tahapan ini adalah sistem operasi Ubuntu 22.04 LTS berhasil terpasang dan berjalan dengan stabil.

3.1.3 Setup Lingkungan Python

Setelah sistem operasi berhasil diinstal, langkah selanjutnya adalah menyiapkan lingkungan pengembangan Python. Tahapan yang dilakukan antara lain:

1. Menginstal Python dan *package manager* pip dengan perintah:

   sudo apt install python3 python3-pip

2. Memastikan versi Python yang digunakan:

   python3 --version

3. Menginstal library psutil yang digunakan untuk mengakses informasi sistem:

   pip3 install psutil

Setelah tahapan ini selesai, lingkungan pengembangan dinyatakan siap digunakan untuk implementasi aplikasi monitoring.

3.2 Implementasi Minggu ke-13: Pengembangan Modul Monitoring

Pada minggu ke-13, fokus kegiatan adalah pengembangan seluruh modul monitoring sesuai dengan perancangan arsitektur yang telah dibuat pada Bab II.

3.2.1 Implementasi Modul Utama (monitor.py)

File monitor.py berfungsi sebagai pusat kendali aplikasi. Modul ini bertugas memanggil seluruh modul monitoring, mengatur interval pembaruan data, serta menampilkan informasi secara terstruktur di terminal.

Pada tahap awal pengujian, interval pembaruan (*refresh rate*) ditetapkan setiap 2 detik. Namun, setelah dilakukan evaluasi, interval ini diubah menjadi 5 detik agar tampilan lebih nyaman dibaca dan penggunaan sumber daya sistem menjadi lebih efisien.

3.2.2 Implementasi Monitoring CPU

Modul cpu_monitor.py dikembangkan untuk menampilkan informasi penggunaan CPU secara keseluruhan dan per-core. Informasi disajikan dalam bentuk persentase dan *progress bar* ASCII sehingga memudahkan pengguna dalam memahami tingkat beban prosesor.

Data penggunaan CPU diperoleh melalui fungsi yang disediakan oleh library psutil. Penggunaan interval pada pemanggilan data dilakukan untuk menghindari nilai 0% pada pemanggilan pertama.

3.2.3 Implementasi Monitoring RAM

Modul ram_monitor.py bertugas menampilkan penggunaan memori sistem. Informasi yang ditampilkan meliputi total RAM, RAM terpakai, memori bebas, buffer, cache, serta swap.

Agar lebih mudah dipahami, ukuran memori diformat ke dalam satuan KB, MB, atau GB menggunakan fungsi bantuan dari modul utils.py. Tampilan *progress bar* digunakan untuk menunjukkan persentase penggunaan RAM.

3.2.4 Implementasi Monitoring Disk

Modul disk_monitor.py digunakan untuk memantau penggunaan media penyimpanan pada setiap partisi disk. Informasi yang ditampilkan meliputi kapasitas total, ruang terpakai, dan ruang kosong.

Untuk menghindari error akibat keterbatasan izin akses pada beberapa partisi, modul ini dilengkapi dengan mekanisme *error handling* sehingga aplikasi tetap berjalan dengan normal.

3.2.5 Implementasi Monitoring Proses

Modul process_monitor.py menampilkan daftar proses yang sedang berjalan dengan penggunaan CPU dan memori tertinggi. Informasi ini berguna untuk mengidentifikasi proses yang berpotensi membebani sistem.

Proses ditampilkan secara terurut berdasarkan tingkat penggunaan sumber daya, disertai grafik ASCII horizontal untuk memperjelas perbandingan antar proses.

3.3 Implementasi Minggu ke-14: Pengujian dan Optimasi

3.3.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk memastikan seluruh fitur aplikasi berjalan sesuai dengan kebutuhan. Pengujian dilakukan dengan membandingkan hasil monitoring aplikasi dengan perintah bawaan Linux seperti top, htop, dan free.

Hasil pengujian menunjukkan bahwa seluruh modul monitoring dapat menampilkan data dengan benar dan konsisten.

3.3.2 Stress Testing

Stress testing dilakukan untuk menguji kestabilan aplikasi dalam kondisi beban sistem yang tinggi. Pengujian dilakukan dengan menjalankan proses yang membebani CPU dan memori secara bersamaan.

Hasil pengujian menunjukkan bahwa aplikasi tetap berjalan stabil, dengan penggunaan CPU aplikasi di bawah 2% dan penggunaan memori sekitar 25 MB.

3.3.3 Perbaikan Bug dan Optimasi

Selama proses pengujian, beberapa permasalahan ditemukan dan diperbaiki, antara lain:

1. Nilai penggunaan CPU bernilai 0% pada pemanggilan pertama.
2. Tampilan *progress bar* tidak muncul pada beberapa jenis terminal.
3. Terjadi *permission denied* pada beberapa partisi disk.

Solusi yang diterapkan meliputi penambahan interval pengambilan data CPU, penggunaan tampilan ASCII cadangan, serta penerapan *try-except* pada modul disk. Selain itu, dilakukan optimasi kode untuk meningkatkan keterbacaan dan efisiensi tanpa mengurangi fungsionalitas sistem.

3.4 Kesimpulan Implementasi

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi *Monitoring Resource System* berhasil dikembangkan dan berjalan sesuai dengan perancangan. Sistem mampu memantau sumber daya komputer secara *real-*

time, stabil, dan akurat, serta siap digunakan sebagai alat bantu monitoring pada sistem operasi Linux.

BAB IV

PENUTUP

4.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, serta pengujian yang telah dilakukan pada proyek *Enhanced Resource Monitoring System*, dapat diambil beberapa kesimpulan sebagai berikut:

1. Aplikasi *Enhanced Resource Monitoring System* berhasil dikembangkan dan dapat berjalan dengan baik pada sistem operasi Linux Ubuntu 22.04 LTS.
2. Penerapan arsitektur modular dengan prinsip *separation of concerns* terbukti memudahkan proses pengembangan, pengujian, dan pemeliharaan aplikasi.
3. Seluruh modul monitoring, yaitu CPU, RAM, disk, dan proses, mampu menampilkan informasi penggunaan sumber daya sistem secara *real-time* dengan tampilan yang informatif dan mudah dipahami.
4. Penggunaan library psutil memungkinkan pengambilan data sistem yang akurat dan efisien, dengan hasil monitoring yang konsisten jika dibandingkan dengan tools bawaan Linux seperti top, htop, dan free.
5. Hasil pengujian fungsional menunjukkan bahwa seluruh fitur aplikasi berjalan sesuai dengan kebutuhan tanpa kesalahan fungsional.
6. Hasil *stress testing* membuktikan bahwa aplikasi tetap stabil meskipun sistem berada dalam kondisi beban tinggi, dengan penggunaan sumber daya aplikasi yang relatif rendah.
7. Secara keseluruhan, aplikasi yang dikembangkan telah memenuhi tujuan proyek dan layak digunakan sebagai alat bantu monitoring sumber daya pada sistem Linux.

4.2 Saran

Meskipun aplikasi *Enhanced Resource Monitoring System* telah berjalan dengan baik, masih terdapat beberapa hal yang dapat dikembangkan lebih lanjut. Adapun saran untuk pengembangan selanjutnya adalah sebagai berikut:

1. Menambahkan antarmuka berbasis grafis (*Graphical User Interface* / GUI) agar aplikasi dapat digunakan oleh pengguna non-teknis.
2. Mengembangkan fitur pencatatan (*logging*) untuk menyimpan riwayat penggunaan sumber daya sistem sehingga dapat dilakukan analisis performa jangka panjang.
3. Menambahkan fitur notifikasi atau peringatan (*alert*) apabila penggunaan CPU, RAM, atau disk melebihi batas tertentu.
4. Mengintegrasikan aplikasi dengan sistem monitoring jarak jauh atau berbasis jaringan untuk memantau beberapa server secara bersamaan.
5. Mengoptimalkan tampilan visualisasi agar lebih adaptif terhadap berbagai ukuran terminal dan lingkungan sistem.

Dengan adanya pengembangan lanjutan tersebut, diharapkan aplikasi ini dapat menjadi sistem monitoring yang lebih lengkap dan bermanfaat dalam lingkungan produksi.

LAMPIRAN

LAMPIRAN A

Struktur Direktori dan Instalasi Python

```
gita@DESKTOP-RUU77IK:/mnt/c/Users/lenovo$ sudo apt install python3 python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
python3 set to manually installed.
```

LAMPIRAN B

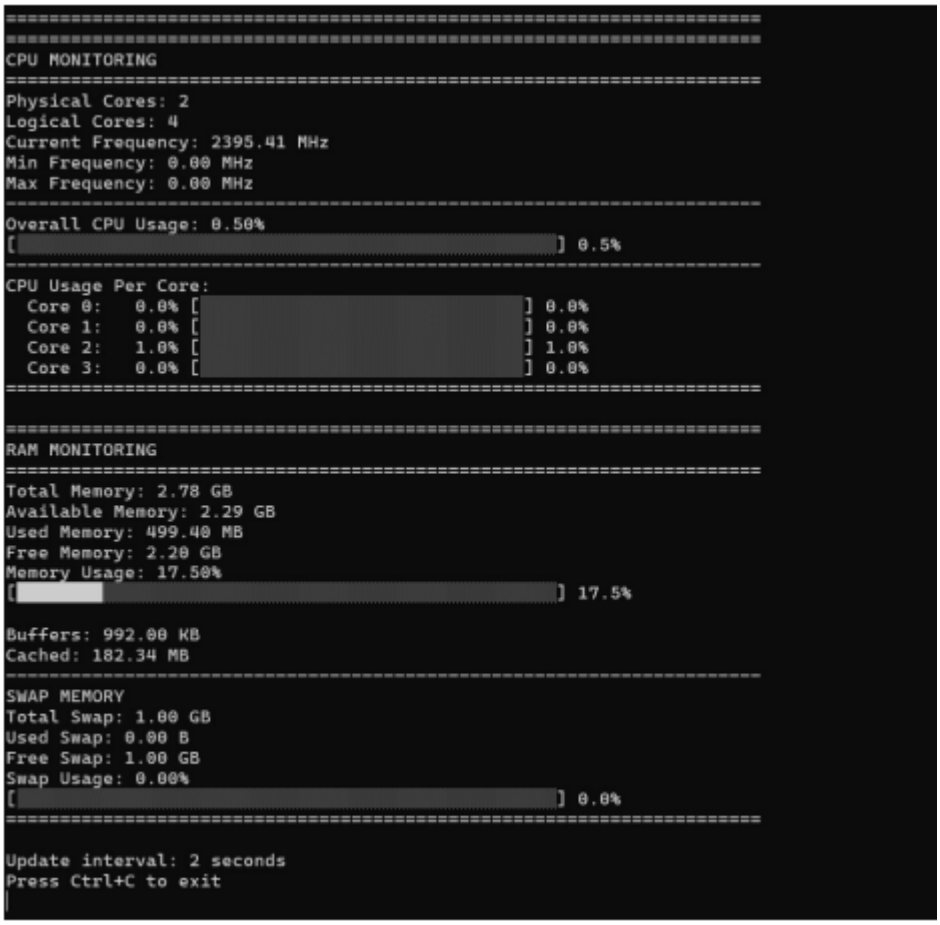
Verifikasi Instalasi pip

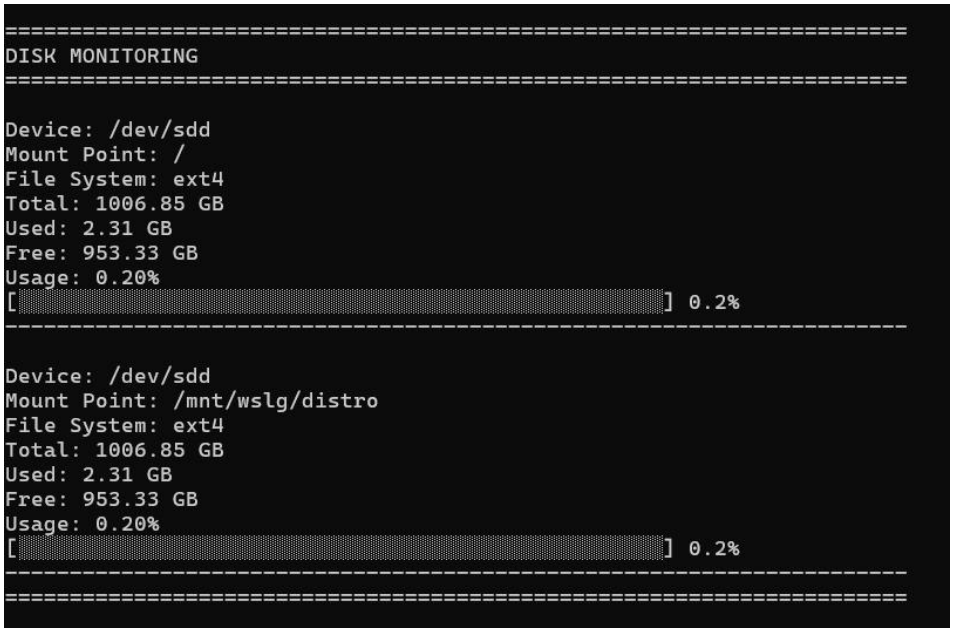
```
gita@DESKTOP-RUU77IK:/mnt/c/Users/lenovo$ pip3 --version
Command 'pip3' not found, but can be installed with:
sudo apt install python3-pip
```

```
gita@DESKTOP-RUU77IK:/mnt/c/Users/lenovo$ sudo apt install python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libdrm-nouveau2 libdrm-radeon1 libgl1-amber-dri
  libglapi-mesa libllvm17t64 libxcb-dri2-0
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  build-essential bzip2 cpp cpp-13 cpp-13-x86-64-linux-gnu
  cpp-x86-64-linux-gnu dpkg-dev fakeroot g++ g++-13
  g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu gcc gcc-13
  gcc-13-base gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu
  javascript-common libalgorithm-diff-perl
  libalgorithm-diff-xs-perl libalgorithm-merge-perl libaom3
  libasan8 libatomic1 libc-dev-bin libc-devtools libc6-dev
  libcc1-0 libcrypt-dev libde265-0 libdpkg-perl libexpat1-dev
  libfakeroot libfile-fcntllock-perl libgcc-13-dev libgd3
  libgomp1 libheif-plugin-aomdec libheif-plugin-aomenc
  libheif-plugin-libde265 libheif1 libhwasan0 libisl23 libitm1
  libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0
  libmpc3 libpython3-dev libpython3.12-dev libquadmath0
  libstdc++-13-dev libtsan2 libubsan1 libxpm4 linux-libc-dev
  lto-disabled-list make manpages-dev python3-dev
  python3-wheel python3.12-dev rpcsvc-proto zlib1g-dev
```

LAMPIRAN C

Contoh Output Program Monitoring


CPU MONITORING
Physical Cores: 2
Logical Cores: 4
Current Frequency: 2395.41 MHz
Min Frequency: 0.00 MHz
Max Frequency: 0.00 MHz
Overall CPU Usage: 0.50%
] 0.5%
CPU Usage Per Core:
Core 0: 0.0% [ ] 0.0%
Core 1: 0.0% [ ] 0.0%
Core 2: 1.0% [ ] 1.0%
Core 3: 0.0% [ ] 0.0%
RAM MONITORING
Total Memory: 2.78 GB
Available Memory: 2.29 GB
Used Memory: 499.40 MB
Free Memory: 2.20 GB
Memory Usage: 17.50%
] 17.5%
Buffers: 992.00 KB
Cached: 182.34 MB
SWAP MEMORY
Total Swap: 1.00 GB
Used Swap: 0.00 B
Free Swap: 1.00 GB
Swap Usage: 0.00%
] 0.0%
Update interval: 2 seconds
Press Ctrl+C to exit


DISK MONITORING
Device: /dev/sdd
Mount Point: /
File System: ext4
Total: 1006.85 GB
Used: 2.31 GB
Free: 953.33 GB
Usage: 0.20%
] 0.2%
Device: /dev/sdd
Mount Point: /mnt/wslg/distro
File System: ext4
Total: 1006.85 GB
Used: 2.31 GB
Free: 953.33 GB
Usage: 0.20%
] 0.2%

```
================================================================================
TOP 5 PROCESSES BY CPU USAGE
================================================================================
PID        NAME                      CPU %      MEM %
--------------------------------------------------------------------------------
1          systemd                   0.00       0.42
2          init-systemd(Ub           0.00       0.07
7          init                      0.00       0.06
71         systemd-journald          0.00       0.64
121        systemd-udevd             0.00       0.21
================================================================================

CPU USAGE GRAPH:
--------------------------------------------------------------------------------
systemd         |  0.0%
init-systemd(Ub |  0.0%
init            |  0.0%
systemd-journal |  0.0%
systemd-udevd   |  0.0%
```