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

BAB I
PENDAHULUAN

1.1 Latar Belakang

Monitoring resource sistem merupakan bagian penting dalam manajemen sistem operasi, khususnya pada sistem operasi Linux. Administrator sistem perlu mengetahui kondisi CPU, RAM, dan proses yang berjalan agar sistem tetap stabil dan optimal. Oleh karena itu, pada proyek ini dikembangkan aplikasi *Monitoring Resource System* berbasis Linux menggunakan bahasa Python dan library **psutil** untuk menampilkan informasi penggunaan resource secara *real-time*.

1.2 Tujuan

Tujuan dari objek ini adalah:

1. Mengembangkan aplikasi monitoring resource berbasis Linux.
2. Menampilkan penggunaan CPU (overall dan per-core).
3. Menampilkan penggunaan RAM dan swap memory.
4. Menyediakan tampilan *real-time* dengan interval update tertentu.
5. Memberikan visualisasi sederhana berupa *progress bar*.

1.3 Manfaat

Manfaat yang diprtoleh dari proyek ini:

1. Menambah pemahaman praktis tentang sistem operasi Linux.
2. Melatih kemampuan pemrograman Python untuk sistem monitoring.
3. Membantu proses monitoring dan troubleshooting sistem.

BAB II
LANDASAN TEORI

2.1 Sistem Operasi Linux

Linux adalah sistem operasi *open-source* yang banyak digunakan pada server dan sistem embedded. Linux menyediakan berbagai antarmuka untuk mengakses informasi sistem, seperti melalui filesystem /proc dan /sys.

2.2 CPU dan Memory

CPU (*Central Processing Unit*) merupakan komponen utama pemrosesan data. Penggunaan CPU biasanya dinyatakan dalam persentase. Memory (RAM) berfungsi menyimpan data dan proses yang sedang berjalan, sedangkan swap digunakan sebagai memory cadangan ketika RAM hampir penuh.

2.3 Library Psutil

Psutil adalah library Python yang digunakan untuk mengambil informasi sistem dan proses, seperti penggunaan CPU, memory, disk, dan proses yang sedang berjalan dengan API yang sederhana.

BAB III
METODOLOGI

3. 1 Timeline Proyek

Pelaksanaan proyek dilakukan sesuai tahapan berikut:

1. Minggu 11 : Penyusunan proposal
2. Minggu 12 : Setup Linux dan Python
3. Minggu 13 : Implementasi monitoring CPU dan RAM
4. Minggu 14 : Pengujian dan optimasi
5. Minggu 15 : Finalisasi dan demo aplikasi

3. 2 Tools yang Digunakan

Perangkat dan tools yang digunakan dalam proyek ini adalah:

1. Sistem Operasi : Ubuntu 22.04 LTS
2. Bahasa Pemrograman : Python 3.10
3. Library : psutil 5.9.5
4. Editor : Visual Studio Code

BAB IV
IMPLEMENTASI

4. 1 Setup Environment

Pada tahap ini dilakukan instalasi Ubuntu, Python, serta library psutil. Setelah itu dilakukan pengujian sederhana untuk memastikan environment siap digunakan.

4. 2 Struktur Project

Aplikasi terdiri dari beberapa file Python yang memiliki fungsi masing-masing. File monitor.py berfungsi sebagai program utama, sedangkan file lainnya berperan sebagai modul pendukung

4. 3 Penjelasan Modul

1. **monitor.py** : Mengatur alur utama program dan interval update.

2. **cpu_monitor.py** : Mengambil dan menampilkan informasi CPU.

3. **ram_monitor.py** : Mengambil dan menampilkan informasi RAM dan swap.

4. **utils.py** : Berisi fungsi bantu seperti *progress bar*, format data, dan pembersihan layar.

4. 4 Alur Kerja Program

Program dijalankan melalui monitor.py, kemudian sistem akan mengambil data CPU dan memory menggunakan psutil, memproses data, dan menampilkannya ke terminal secara *real-time* hingga pengguna menekan Ctrl + C.

4. 5 Pengujian Sistem

Pengujian dilakukan dengan kondisi idle dan kondisi stress test untuk memastikan data yang ditampilkan sesuai dengan kondisi sistem.

BAB V
HASIL DAN ANALISIS

5.1 Hasil Pengujian

Hasil pengujian menunjukkan bahwa aplikasi mampu menampilkan penggunaan CPU dan RAM secara akurat dan stabil.

5.2 Analisa Peforma

Aplikasi memiliki penggunaan resource yang ringan dengan CPU overhead kurang dari 1% dan penggunaan memory sekitar 15–20 MB.

5.3 Kelebihan dan kekurangan

Kelebihan:

a. Ringan dan efisien

b. Data akurat

c. Struktur kode modular

Keterbatasan:

a. Tampilan berbasis CLI

b. Belum mendukung penyimpanan data historis

c. Khusus untuk Linux

BAB VI
PENITUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa aplikasi **Monitoring Resource System** berbasis Linux berhasil dikembangkan dan berjalan sesuai dengan tujuan yang telah ditetapkan. Aplikasi ini mampu menampilkan informasi penggunaan sumber daya sistem, khususnya CPU dan RAM, secara *real-time* dengan tampilan yang sederhana dan mudah dipahami oleh pengguna.Seluruh fitur utama yang dirancang, seperti monitoring CPU (overall dan per-core), monitoring penggunaan RAM dan swap memory, pembaruan data secara otomatis dengan interval tertentu, serta visualisasi penggunaan resource dalam bentuk *progress bar*, telah berfungsi dengan baik tanpa mengalami kendala yang berarti. Hasil pengujian pada kondisi sistem normal maupun saat dilakukan *stress testing* menunjukkan bahwa aplikasi dapat mendeteksi perubahan penggunaan resource dengan akurat dan responsif.Selain itu, aplikasi ini memiliki performa yang ringan dengan penggunaan CPU dan memory yang relatif kecil, sehingga tidak mengganggu kinerja sistem yang sedang dimonitor. Struktur kode yang modular juga mempermudah proses pengembangan, pemeliharaan, serta pengembangan fitur di masa mendatang. Dengan demikian, aplikasi Monitoring Resource System ini dapat dijadikan sebagai alat bantu monitoring sistem Linux sekaligus sarana pembelajaran yang efektif dalam memahami konsep manajemen sumber daya pada sistem operasi.

6.2 Saran

Meskipun aplikasi Monitoring Resource System telah berjalan dengan baik, masih terdapat beberapa pengembangan yang dapat dilakukan agar aplikasi menjadi lebih optimal dan fungsional. Salah satu pengembangan yang disarankan adalah penambahan **antarmuka grafis (GUI)** agar aplikasi lebih ramah pengguna, terutama bagi pengguna yang kurang terbiasa dengan tampilan berbasis *command line interface* (CLI).Selain itu, aplikasi dapat dikembangkan dengan menambahkan fitur **logging atau penyimpanan data historis**, sehingga pengguna dapat melihat riwayat penggunaan CPU dan memory dalam jangka waktu

tertentu. Fitur ini akan sangat berguna untuk analisis performa sistem dan keperluan troubleshooting. Pengembangan lainnya adalah penambahan fitur monitoring **disk usage dan network activity**, sehingga informasi yang ditampilkan menjadi lebih lengkap dan menyeluruh.Untuk pengembangan lebih lanjut, aplikasi juga dapat dibuat agar bersifat **cross-platform**, sehingga tidak hanya berjalan pada sistem operasi Linux, tetapi juga dapat digunakan pada sistem operasi lain seperti Windows dan macOS. Dengan adanya pengembangan tersebut, diharapkan aplikasi Monitoring Resource System dapat menjadi lebih bermanfaat, fleksibel, dan sesuai dengan kebutuhan pengguna di berbagai lingkungan sistem.

LAMPIRAN

A. Source Code

```
import os

import signal

import psutil

from typing import Optional, Tuple

class ProcessKiller:

"""Class untuk menangani terminasi proses"""

def __init__(self):

self.last_error = None

def get_process_info(self, pid: int) -> Optional[dict]:

"""

Mendapatkan informasi proses berdasarkan PID

Args:

pid: Process ID

Returns:

Dictionary berisi informasi proses atau None jika error

"""

try:
```

```
process = psutil.Process(pid)

return {

'pid': pid,

'name': process.name(),

'username': process.username(),

'status': process.status(),

'cpu_percent': process.cpu_percent(),

'memory_percent': process.memory_percent()

}

except psutil.NoSuchProcess:

self.last_error = f"Process with PID {pid} does not exist"

return None

except psutil.AccessDenied:

self.last_error = f"Access denied to process {pid}"

return None

def kill_process_graceful(self, pid: int) -> Tuple[bool, str]:

"""

Menghentikan proses dengan SIGTERM (graceful shutdown)

SIGTERM memberikan kesempatan kepada proses untuk: - Menutup file yang terbuka - Membersihkan resources - Melakukan cleanup operations
```

```
Args:

pid: Process ID yang akan dihentikan

Returns:

Tuple (success: bool, message: str)

"""

try:

# Cek apakah proses ada

process = psutil.Process(pid)

process_name = process.name()

# Kirim SIGTERM (signal 15)

os.kill(pid, signal.SIGTERM)

# Tunggu proses terminate (max 5 detik)

try:

process.wait(timeout=5)

return (True, f"Process {process_name} (PID: {pid}) terminated

successfully")

except psutil.TimeoutExpired:

return (False, f"Process {process_name} (PID: {pid}) did not

respond to SIGTERM")
```

```
except psutil.NoSuchProcess:

return (False, f"Process with PID {pid} does not exist")

except psutil.AccessDenied:

return (False, f"Permission denied. Cannot kill process {pid} (try

with sudo)")

except Exception as e:

return (False, f"Error killing process: {str(e)}")

def kill_process_force(self, pid: int) -> Tuple[bool, str]:

"""

Menghentikan proses dengan SIGKILL (force kill)

SIGKILL tidak dapat di-handle atau di-ignore oleh proses.

Gunakan hanya jika SIGTERM gagal.

Args:

pid: Process ID yang akan dihentikan

Returns:

Tuple (success: bool, message: str)

"""

try:

# Cek apakah proses ada
```

```
process = psutil.Process(pid)

process_name = process.name()

# Kirim SIGKILL (signal 9)

os.kill(pid, signal.SIGKILL)

# Verifikasi proses terminated

try:

process.wait(timeout=2)

return (True, f"Process {process_name} (PID: {pid}) force

killed successfully")

except psutil.TimeoutExpired:

return (False, f"Failed to kill process {process_name} (PID:

{pid})")

except psutil.NoSuchProcess:

return (False, f"Process with PID {pid} does not exist")

except psutil.AccessDenied:

return (False, f"Permission denied. Cannot kill process {pid} (try

with sudo)")

except Exception as e:

return (False, f"Error force killing process: {str(e)}")
```

```
def kill_process_interactive(self, pid: int, force: bool = False) ->

Tuple[bool, str]:

"""

Kill process dengan konfirmasi user

Args:

pid: Process ID

force: True untuk SIGKILL, False untuk SIGTERM

Returns:

Tuple (success: bool, message: str)

"""

# Dapatkan info proses

info = self.get_process_info(pid)

if not info:

return (False, self.last_error)

# Tampilkan informasi proses

print(f"\n{'='*50}")

print(f"Process Information:")

print(f"  PID: {info['pid']}")

print(f"  Name: {info['name']}")
```

```
print(f"  User: {info['username']}")

print(f"  Status: {info['status']}")

print(f"  CPU: {info['cpu_percent']:.1f}%")

print(f"  Memory: {info['memory_percent']:.1f}%")

print(f"{'='*50}")

# Konfirmasi

signal_type = "SIGKILL (force)" if force else "SIGTERM (graceful)"

confirm = input(f"Kill this process with {signal_type}? (yes/no):

").lower()

if confirm != 'yes':

return (False, "Operation cancelled by user")

# Eksekusi kill

if force:

return self.kill_process_force(pid)

else:

return self.kill_process_graceful(pid)

# Fungsi utility untuk monitoring

def get_running_processes():
```

B. Hasil Pengujian

```
(venv) gita@DESKTOP-RUU77IK:~/MRS$ top
top - 16:41:01 up  3:36,  1 user,  load average: 0.00, 0.02, 0.00
Tasks:  33 total,   1 running,  32 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.2 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   2847.2 total,   2265.8 free,    465.2 used,    203.5 buff/cache
MiB Swap:   1024.0 total,   1024.0 free,      0.0 used.   2382.1 avail Mem
```

```
Processing triggers for man-db (2.12.0-4build2) ...
(venv) gita@DESKTOP-RUU77IK:~/MRS$ stress --vm 1 --vm-bytes 1G --timeout 30s
stress: info: [6009] dispatching hogs: 0 cpu, 0 io, 1 vm, 0 hdd
stress: info: [6009] successful run completed in 31s
(venv) gita@DESKTOP-RUU77IK:~/MRS$
```

C. Cara Penggunaan

- Instal library: pip3 install psutil
- Jalankan program: python3 monitor.py
- Hentikan program: Ctrl + C