LAPORAN PROGRESS 2(MINGGU 13)

Pengecekan Phyton dan Instalasi Library Monitoring

PROJECT 3

1. Tujuan

Tujuan dari progress 2 adalah mengimplementasikan fitur monitoring CPU dan RAM secara real-time menggunakan library prutil yang telah diinstal pada progress 1. Program dirancang untuk menampilkan informasi penggunaan CPU(total dan per-core) serta penggunaan RAM(total, used, free, dan presentance) dengan sistem ring buffer untuk menyimpan histori 10 data terakhir.

2. Langkah-langkah pelaksanaan

2.1 Import Library yang Diperlukan

pada bagian awal program, dilakukan import library yang dibutuhkan untuk monitoring sistem

```
 GNU nano 7.2                                                    linux_task
# Linux Task Manager (Monitoring CPU & RAM)
# Project 3 - Sistem Operasi
# Minggu 13 : Implementasi Monitoring Resource

import psutil                  # Library untuk monitoring resource sistem
import time                    # Untuk pengaturan delay waktu
import os                      # Untuk perintah sistem (clear terminal)
from collections import deque  # Struktur data ring buffer
```

2.2 Inisialisasi Ring Buffer

Membuat ring buffer menggunakan deque dengan maksimal 10 data untuk menyimpan histori CPU dan RAM

```
# Ring buffer untuk meniyimpan histori CPU dan RAM dan Digunakan sebagai persiapan pembuatan grafik
cpu_history = deque(maxlen=10) # Menyimpan 10 data CPU terakhir
ram_history = deque(maxlen=10) # Menyimpan 10 data RAM terakhir
```

2.3 Implementasi Loop Monitoring Real-Time

Membuat loop utama yang terus berjalan untuk monitoring

```
# Loop utama monitoring (real-time)
while True:
        os.system("clear") # Membersihkan tampilan terminal setiap iterasi
```

2.4 CPU Monitoring

A. Monitoring CPU Total

```
# CPU Monitoring
cpu_total = psutil.cpu_percent(interval=1) # Mengambil pesentase penggunaan CPU total
cpu_per_core = psutil.cpu_percent(percpu=True) # Mengambil persentase penggunaan CPU per core

# RAM Monitoring
mem = psutil.virtual_memory() # Mengambil informasi penggunaan memori

# Menyimpan data ke dalam ring buffer
cpu_history.append(cpu_total)
ram_history.append(mem.percent)

# Output ke terminal
print("=== MONITORING CPU ===")
print(f"CPU Total : {cpu_total}%")
```

B. Menghitung CPU per-core

```
cpu_per_core = psutil.cpu_percent(percpu=True) # Mengambil persentase penggunaan CPU per core

# RAM Monitoring
mem = psutil.virtual_memory() # Mengambil informasi penggunaan memori

# Menyimpan data ke dalam ring buffer
cpu_history.append(cpu_total)
ram_history.append(mem.percent)

# Output ke terminal
print("=== MONITORING CPU ===")
print(f"CPU Total : {cpu_total}%")

# Menampilkan penggunaan CPU tiap core
for i, core in enumerate(cpu_per_core):
        print(f"Core {i} : {core}%")
```

2.5 RAM Monitoring

Mengambil informasi penggunaan memori dan menampilkannya

```
# RAM Monitoring
mem = psutil.virtual_memory() # Mengambil informasi penggunaan memori

# Menyimpan data ke dalam ring buffer
cpu_history.append(cpu_total)
ram_history.append(mem.percent)

# Output ke terminal
print("=== MONITORING CPU ===")
print(f"CPU Total : {cpu_total}%")

# Menampilkan penggunaan CPU tiap core
for i, core in enumerate(cpu_per_core):
        print(f"Core {i} : {core}%")


print("\n=== MONITORING RAM ===")
print(f"Total RAM       : {mem.total / (1024**3):.2f} GB")
print(f"Used RAM        : {mem.used / (1024**3):.2} GB")
print(f"Free RAM        : {mem.available / (1024**3):.2f} GB")
print(f"Usage           : {mem.percent}%")
```

2.6 Penyimpanan Data ke Ring Buffer

Data CPU dan RAM disimpan kedalam ring buffer;

```
# Menyimpan data ke dalam ring buffer
cpu_history.append(cpu_total)
ram_history.append(mem.percent)
```

2.7 Menampilkan History Data

Menampilkan 10 data terakhir yang tersimpan dalam ring buffer

```
# Menampilkan data CPU & RAM terakhir (ring buffer))
print("\n=== HISTORI ( 10 DATA TERAKHIR) ===")
print("CPU History:", list(cpu_history))
print("RAM History:", list(ram_history))
```

2.8 Delay dan Refresh

Memberikan delay 1 detik sebelum ke iterasi berikutnya.

```
# Delay tambahan agar tampilan stabil
time.sleep(1)
```

2.9 Menjalankan Program

```
=== MONITORING CPU ===
CPU Total : 0.1%
Core 0 : 0.0%
Core 1 : 0.0%
Core 2 : 0.0%
Core 3 : 0.0%
Core 4 : 0.0%
Core 5 : 0.0%
Core 6 : 0.5%
Core 7 : 0.0%

=== MONITORING RAM ===
Total RAM       : 3.70 GB
Used RAM        : 0.5 GB
Free RAM        : 3.20 GB
Usage           : 13.5%

=== HISTORI ( 10 DATA TERAKHIR) ===
CPU History: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.1]
RAM History: [13.5, 13.5, 13.5, 13.5, 13.5, 13.5, 13.5, 13.5, 13.5, 13.5]
```

3. Hasil yang Diperoleh

Berdasarkan implementasi yang telah dilakukan pada progress 2, diperoleh hasil sebagai berikut:

1. Program `linux_task_manager.py` berhasil dijalankan tanpa error
2. Monitoring CPU Total berhasil menampilkan persentase penggunaan CPU
3. Monitoring CPU Per Core berhasil menampilkan penggunaan setiap core processor
4. Monitoring RAM berhasil menampilkan Total, Used, Free, dan Usage dalam GB
5. Ring buffer berhasil diimplementasikan menggunakan deque(maxlen=10)
6. History 10 data terakhir CPU dan RAM berhasil ditampilkan dalam format list
7. Terminal auto-refresh setiap 1 detik menggunakan os.system("clear")
8. Program berjalan stabil dalam mode continuous monitoring
9. Format output rapi dan terstruktur dengan informasi yang jelas
10. Semua library (psutil, time, os, collections) berhasil digunakan dengan baik