MONITORANDO A INTEGRIDADE DOS ARQUIVOS PRESERVADOS

O script file_integrity_monitor_multi_dir.py tem como objetivo monitorar a integridade de um ou mais diretórios gerando como resultado um arquivo txt e um html em formato de relatório e envia via e-mail.

```
#!/usr/bin/env python3
"""
File Integrity Monitor
Sistema de monitoramento de integridade de arquivos com suporte a múltiplos diretórios
"""

import os
import sys
import json
import hashlib
import subprocess
from datetime import datetime
from pathlib import Path
import argparse
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

class FileIntegrityMonitor:
    def __init__(self, target_dirs, database_dir=".file_integrity"):
        # Suporta lista de diretórios
        if isinstance(target_dirs, str):
            target_dirs = [target_dirs]
        self.target_dirs = [Path(d).resolve() for d in target_dirs]
        self.database_dir = Path(database_dir)
        self.database_dir.mkdir(exist_ok=True)
        self.current_scan_file = self.database_dir / "current_scan.json"
```

```
    self.previous_scan_file = self.database_dir / "previous_scan.json"
    self.history_file = self.database_dir / "scan_history.json"
    self.config_file = self.database_dir / "config.json"
    self.email_config_file = self.database_dir / "email_config.json"

def calculate_hash(self, filepath, algorithm='sha256'):
    """Calcula hash de um arquivo"""
    hash_obj = hashlib.new(algorithm)
    try:
        with open(filepath, 'rb') as f:
            for chunk in iter(lambda: f.read(4096), b''):
                hash_obj.update(chunk)
        return hash_obj.hexdigest()
    except Exception as e:
        return f"ERROR: {str(e)}"

def scan_directory(self):
    """Escaneia todos os diretórios configurados"""
    print(f"Escaneando {len(self.target_dirs)} diretório(s)...")
    scan_data = {
        'timestamp': datetime.now().isoformat(),
        'target_dirs': [str(d) for d in self.target_dirs],
        'files': {}
    }

    total_files = 0
    for target_dir in self.target_dirs:
        if not target_dir.exists():
            print(f"⚠  Diretório não encontrado: {target_dir}")
            continue

        print(f"\n Escaneando: {target_dir}")
        dir_files = 0

        for root, dirs, files in os.walk(target_dir):
            # Ignora o diretório de database
            if self.database_dir.name in dirs:
                dirs.remove(self.database_dir.name)

            for filename in files:
```

```
                    filepath = Path(root) / filename

                    # Usa caminho relativo ao diretório base
                    try:
                        relative_path = filepath.relative_to(target_dir)
                        # Prefixo com o nome do diretório base para evitar conflitos
                        key = f"{target_dir.name}/{relative_path}"
                    except ValueError:
                        # Se não conseguir fazer relativo, usa caminho absoluto
                        key = str(filepath)

                    try:
                        file_stat = filepath.stat()
                        file_hash = self.calculate_hash(filepath)

                        scan_data['files'][key] = {
                            'hash': file_hash,
                            'size': file_stat.st_size,
                            'modified':
datetime.fromtimestamp(file_stat.st_mtime).isoformat(),
                            'full_path': str(filepath),
                            'base_dir': str(target_dir)
                        }
                        dir_files += 1
                        total_files += 1

                        if dir_files % 100 == 0:
                            print(f"  Processados {dir_files} arquivos...")
                    except Exception as e:
                        print(f"  Erro ao processar {filepath}: {e}")

            print(f"  ✓ {dir_files} arquivos processados")

        print(f"\n✓ Scan completo: {total_files} arquivos no total")
        return scan_data

    def save_scan(self, scan_data):
        """Salva dados do scan"""
        # Move scan atual para anterior
        if self.current_scan_file.exists():
```

```
            self.current_scan_file.rename(self.previous_scan_file)

        # Salva novo scan
        with open(self.current_scan_file, 'w') as f:
            json.dump(scan_data, f, indent=2)

        # Adiciona ao histórico
        self.add_to_history(scan_data)

    def add_to_history(self, scan_data):
        """Adiciona scan ao histórico"""
        history = []
        if self.history_file.exists():
            with open(self.history_file, 'r') as f:
                history = json.load(f)

        history_entry = {
            'timestamp': scan_data['timestamp'],
            'total_files': len(scan_data['files']),
            'total_size': sum(f['size'] for f in scan_data['files'].values()),
            'directories': scan_data['target_dirs']
        }
        history.append(history_entry)

        # Mantém apenas últimos 50 registros
        history = history[-50:]

        with open(self.history_file, 'w') as f:
            json.dump(history, f, indent=2)

    def compare_scans(self):
        """Compara scan atual com anterior"""
        if not self.previous_scan_file.exists():
            return None

        with open(self.current_scan_file, 'r') as f:
            current = json.load(f)

        with open(self.previous_scan_file, 'r') as f:
            previous = json.load(f)
```

```
current_files = set(current['files'].keys())
previous_files = set(previous['files'].keys())

# Arquivos novos
new_files = current_files - previous_files

# Arquivos removidos
removed_files = previous_files - current_files

# Arquivos modificados e corrompidos
modified_files = []
corrupted_files = []
intact_files = []

common_files = current_files & previous_files
for filepath in common_files:
    current_hash = current['files'][filepath]['hash']
    previous_hash = previous['files'][filepath]['hash']

    if current_hash.startswith('ERROR'):
        corrupted_files.append(filepath)
    elif current_hash != previous_hash:
        modified_files.append(filepath)
    else:
        intact_files.append(filepath)

# Agrupa por diretório base
def group_by_dir(file_list):
    by_dir = {}
    for f in file_list:
        base_dir = current['files'].get(f, {}).get('base_dir') or \
                   previous['files'].get(f, {}).get('base_dir', 'unknown')
        if base_dir not in by_dir:
            by_dir[base_dir] = []
        by_dir[base_dir].append(f)
    return by_dir

return {
    'current_timestamp': current['timestamp'],
```

```
            'previous_timestamp': previous['timestamp'],
            'total_current': len(current_files),
            'total_previous': len(previous_files),
            'new_files': list(new_files),
            'removed_files': list(removed_files),
            'modified_files': modified_files,
            'corrupted_files': corrupted_files,
            'intact_files': intact_files,
            'new_by_dir': group_by_dir(new_files),
            'modified_by_dir': group_by_dir(modified_files),
            'corrupted_by_dir': group_by_dir(corrupted_files),
            'target_dirs': current['target_dirs']
        }

    def generate_report(self, comparison):
        """Gera relatório de integridade"""
        if comparison is None:
            print("\n" + "="*60)
            print("PRIMEIRO SCAN - Baseline criado")
            print("="*60)
            print(f"Diretórios monitorados: {len(self.target_dirs)}")
            for d in self.target_dirs:
                print(f"  • {d}")
            return

        total = comparison['total_current']
        intact = len(comparison['intact_files'])
        modified = len(comparison['modified_files'])
        corrupted = len(comparison['corrupted_files'])
        new = len(comparison['new_files'])
        removed = len(comparison['removed_files'])

        # Calcula porcentagens
        intact_pct = (intact / total * 100) if total > 0 else 0
        modified_pct = (modified / total * 100) if total > 0 else 0
        corrupted_pct = (corrupted / total * 100) if total > 0 else 0
        new_pct = (new / total * 100) if total > 0 else 0

        print("\n" + "="*60)
        print("RELATÓRIO DE INTEGRIDADE DE ARQUIVOS")
```

```
print("="*60)
print(f"Scan atual:    {comparison['current_timestamp']}")
print(f"Scan anterior: {comparison['previous_timestamp']}")
print(f"\nDiretórios monitorados: {len(comparison['target_dirs'])}")
for d in comparison['target_dirs']:
    print(f"  • {d}")

print("\n" + "-"*60)
print("ESTATÍSTICAS GERAIS")
print("-"*60)
print(f"Total de arquivos atuais: {total}")
print(f"Total de arquivos anteriores: {comparison['total_previous']}")

print("\n" + "-"*60)
print("GRAU DE INTEGRIDADE")
print("-"*60)
print(f"✓ Arquivos íntegros:    {intact:4d} ({intact_pct:6.2f}%)")
print(f"≠ Arquivos modificados: {modified:4d} ({modified_pct:6.2f}%)")
print(f"✗ Arquivos corrompidos: {corrupted:4d} ({corrupted_pct:6.2f}%)")
print(f"+ Arquivos novos:       {new:4d} ({new_pct:6.2f}%)")
print(f"- Arquivos removidos:   {removed:4d}")

# Integridade geral
if intact + modified + corrupted > 0:
    integrity_score = (intact / (intact + modified + corrupted)) * 100
    print(f"\n{'='*60}")
    print(f"INTEGRIDADE GERAL: {integrity_score:.2f}%")
    print(f"{'='*60}")

# Detalhes por diretório
if corrupted > 0:
    print("\n" + "-"*60)
    print("ARQUIVOS CORROMPIDOS POR DIRETÓRIO")
    print("-"*60)
    for base_dir, files in comparison['corrupted_by_dir'].items():
        print(f"\n  □□{base_dir} ({len(files)} arquivos)")
        for f in files[:10]:
            print(f"    ✗ {f}")
        if len(files) > 10:
            print(f"    ... e mais {len(files) - 10} arquivos")
```

```
        if modified > 0:
            print("\n" + "-"*60)
            print("ARQUIVOS MODIFICADOS POR DIRETÓRIO")
            print("-"*60)
            for base_dir, files in comparison['modified_by_dir'].items():
                print(f"\n  □□{base_dir} ({len(files)} arquivos)")
                for f in files[:10]:
                    print(f"    ≠ {f}")
                if len(files) > 10:
                    print(f"    ... e mais {len(files) - 10} arquivos")

        # Salva relatórios
        self.save_report_to_file(comparison)
        html_file = self.save_html_report(comparison)

        return html_file

    def save_report_to_file(self, comparison):
        """Salva relatório em arquivo texto"""
        report_file = self.database_dir /
f"report_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"

        with open(report_file, 'w') as f:
            f.write("="*60 + "\n")
            f.write("RELATÓRIO DE INTEGRIDADE DE ARQUIVOS\n")
            f.write("="*60 + "\n")
            f.write(f"Scan atual:    {comparison['current_timestamp']}\n")
            f.write(f"Scan anterior: {comparison['previous_timestamp']}\n\n")

            f.write(f"Diretórios monitorados: {len(comparison['target_dirs'])}\n")
            for d in comparison['target_dirs']:
                f.write(f"  • {d}\n")

            total = comparison['total_current']
            intact = len(comparison['intact_files'])
            modified = len(comparison['modified_files'])
            corrupted = len(comparison['corrupted_files'])
            new = len(comparison['new_files'])
            removed = len(comparison['removed_files'])
```

```
intact_pct = (intact / total * 100) if total > 0 else 0
modified_pct = (modified / total * 100) if total > 0 else 0
corrupted_pct = (corrupted / total * 100) if total > 0 else 0

f.write("\n" + "-"*60 + "\n")
f.write("GRAU DE INTEGRIDADE\n")
f.write("-"*60 + "\n")
f.write(f"Arquivos íntegros:    {intact} ({intact_pct:.2f}%)\n")
f.write(f"Arquivos modificados: {modified} ({modified_pct:.2f}%)\n")
f.write(f"Arquivos corrompidos: {corrupted} ({corrupted_pct:.2f}%)\n")
f.write(f"Arquivos novos:       {new}\n")
f.write(f"Arquivos removidos:   {removed}\n\n")

if intact + modified + corrupted > 0:
    integrity_score = (intact / (intact + modified + corrupted)) * 100
    f.write("="*60 + "\n")
    f.write(f"INTEGRIDADE GERAL: {integrity_score:.2f}%\n")
    f.write("="*60 + "\n\n")

# Lista completa de arquivos com problemas por diretório
if corrupted:
    f.write("\nARQUIVOS CORROMPIDOS POR DIRETÓRIO:\n")
    for base_dir, files in comparison['corrupted_by_dir'].items():
        f.write(f"\n{base_dir}:\n")
        for file in files:
            f.write(f"  {file}\n")

if modified:
    f.write("\nARQUIVOS MODIFICADOS POR DIRETÓRIO:\n")
    for base_dir, files in comparison['modified_by_dir'].items():
        f.write(f"\n{base_dir}:\n")
        for file in files:
            f.write(f"  {file}\n")

if new:
    f.write("\nARQUIVOS NOVOS POR DIRETÓRIO:\n")
    for base_dir, files in comparison['new_by_dir'].items():
        f.write(f"\n{base_dir}:\n")
        for file in files:
```

```
                        f.write(f"  {file}\n")

        print(f"\nRelatório TXT salvo em: {report_file}")

    def save_html_report(self, comparison):
        """Gera relatório HTML"""
        timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
        report_file = self.database_dir / f"report_{timestamp}.html"

        total = comparison['total_current']
        intact = len(comparison['intact_files'])
        modified = len(comparison['modified_files'])
        corrupted = len(comparison['corrupted_files'])
        new = len(comparison['new_files'])
        removed = len(comparison['removed_files'])

        intact_pct = (intact / total * 100) if total > 0 else 0
        modified_pct = (modified / total * 100) if total > 0 else 0
        corrupted_pct = (corrupted / total * 100) if total > 0 else 0
        new_pct = (new / total * 100) if total > 0 else 0

        integrity_score = (intact / (intact + modified + corrupted)) * 100 if (intact +
modified + corrupted) > 0 else 0

        # Determina cor e status
        if integrity_score >= 99:
            status_color = "#28a745"
            status_text = "EXCELENTE"
        elif integrity_score >= 95:
            status_color = "#5bc0de"
            status_text = "BOM"
        elif integrity_score >= 90:
            status_color = "#ffc107"
            status_text = "ATENÇÃO"
        else:
            status_color = "#dc3545"
            status_text = "CRÍTICO"

        # Lista de diretórios
        dirs_html = ""
```

```
        for d in comparison['target_dirs']:
            dirs_html += f"<li>{d}</li>"

        html_content = f"""
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Relatório de Integridade - {timestamp}</title>
    <style>
        * {{ margin: 0; padding: 0; box-sizing: border-box; }}
        body {{ font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif; background:
linear-gradient(135deg, #667eea 0%, #764ba2 100%); padding: 20px; color: #333; }}
        .container {{ max-width: 1200px; margin: 0 auto; background: white; border-radius:
10px; box-shadow: 0 10px 40px rgba(0,0,0,0.2); overflow: hidden; }}
        .header {{ background: linear-gradient(135deg, #667eea 0%, #764ba2 100%); color:
white; padding: 40px; text-align: center; }}
        .header h1 {{ font-size: 2.5em; margin-bottom: 10px; }}
        .header .subtitle {{ opacity: 0.9; font-size: 1.1em; }}
        .integrity-score {{ background: {status_color}; color: white; padding: 40px; text-
align: center; font-size: 3em; font-weight: bold; }}
        .integrity-score .label {{ font-size: 0.4em; opacity: 0.9; display: block; margin-
bottom: 10px; }}
        .integrity-score .status {{ font-size: 0.5em; margin-top: 10px; }}
        .content {{ padding: 40px; }}
        .directories {{ background: #f8f9fa; padding: 20px; border-radius: 8px; margin-bottom:
30px; }}
        .directories h3 {{ color: #667eea; margin-bottom: 15px; }}
        .directories ul {{ list-style: none; }}
        .directories li {{ padding: 8px; margin: 5px 0; background: white; border-radius: 4px;
border-left: 3px solid #667eea; }}
        .stats-grid {{ display: grid; grid-template-columns: repeat(auto-fit, minmax(250px,
1fr)); gap: 20px; margin-bottom: 40px; }}
        .stat-card {{ background: white; padding: 25px; border-radius: 8px; border: 2px solid
#e9ecef; transition: transform 0.2s; }}
        .stat-card:hover {{ transform: translateY(-5px); box-shadow: 0 5px 20px
rgba(0,0,0,0.1); }}
        .stat-card.intact {{ border-left: 4px solid #28a745; }}
        .stat-card.modified {{ border-left: 4px solid #ffc107; }}
```

```
        .stat-card.corrupted {{ border-left: 4px solid #dc3545; }}
        .stat-card .number {{ font-size: 2.5em; font-weight: bold; margin: 10px 0; }}
        .stat-card .percentage {{ color: #666; font-size: 1.2em; }}
        .file-list {{ background: #f8f9fa; border-radius: 8px; padding: 20px; margin-top:
20px; }}
        .file-list h3 {{ color: #333; margin-bottom: 15px; padding-bottom: 10px; border-
bottom: 2px solid #dee2e6; }}
        .file-list .dir-section {{ margin: 20px 0; }}
        .file-list .dir-title {{ font-weight: bold; color: #667eea; margin-bottom: 10px; }}
        .file-list ul {{ list-style: none; max-height: 300px; overflow-y: auto; }}
        .file-list li {{ padding: 8px; margin: 5px 0; background: white; border-radius: 4px;
word-break: break-all; }}
        .file-list.corrupted li {{ border-left: 3px solid #dc3545; }}
        .file-list.modified li {{ border-left: 3px solid #ffc107; }}
        .timestamp {{ background: #e9ecef; padding: 15px; border-radius: 5px; margin-bottom:
20px; }}
        .timestamp strong {{ color: #667eea; }}
        .footer {{ background: #f8f9fa; padding: 20px; text-align: center; color: #666; font-
size: 0.9em; }}
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>□□Relatório de Integridade de Arquivos</h1>
            <div class="subtitle">Monitoramento de Múltiplos Diretórios</div>
        </div>

        <div class="integrity-score">
            <span class="label">INTEGRIDADE GERAL</span>
            {integrity_score:.2f}%
            <div class="status">{status_text}</div>
        </div>

        <div class="content">
            <div class="timestamp">
                <strong>Scan Atual:</strong> {comparison['current_timestamp']}<br>
                <strong>Scan Anterior:</strong> {comparison['previous_timestamp']}
            </div>
```

```
            <div class="directories">
                <h3>□□Diretórios Monitorados ({len(comparison['target_dirs'])})</h3>
                <ul>{dirs_html}</ul>
            </div>

            <h2 style="margin: 30px 0 20px 0; color: #667eea;">Estatísticas Globais</h2>

            <div class="stats-grid">
                <div class="stat-card intact">
                    <div style="font-size:2em">✓</div>
                    <div class="number">{intact}</div>
                    <div class="percentage">{intact_pct:.2f}%</div>
                    <div>Arquivos Íntegros</div>
                </div>

                <div class="stat-card modified">
                    <div style="font-size:2em">≠</div>
                    <div class="number">{modified}</div>
                    <div class="percentage">{modified_pct:.2f}%</div>
                    <div>Arquivos Modificados</div>
                </div>

                <div class="stat-card corrupted">
                    <div style="font-size:2em">✗</div>
                    <div class="number">{corrupted}</div>
                    <div class="percentage">{corrupted_pct:.2f}%</div>
                    <div>Arquivos Corrompidos</div>
                </div>
            </div>
"""

        # Adiciona seções por diretório
        if corrupted > 0:
            html_content += '<div class="file-list corrupted"><h3>⚠□ Arquivos
Corrompidos</h3>'
            for base_dir, files in comparison['corrupted_by_dir'].items():
                html_content += f'<div class="dir-section"><div class="dir-title">□□{base_dir}
({len(files)})</div><ul>'
                for f in files[:50]:
                    html_content += f"<li>✗ {f}</li>"
```

```
                if len(files) > 50:
                    html_content += f"<li><strong>... e mais {len(files) - 50}
arquivos</strong></li>"
                html_content += '</ul></div>'
            html_content += '</div>'

        if modified > 0:
            html_content += '<div class="file-list modified"><h3>□□Arquivos Modificados</h3>'
            for base_dir, files in comparison['modified_by_dir'].items():
                html_content += f'<div class="dir-section"><div class="dir-title">□□{base_dir}
({len(files)})</div><ul>'
                for f in files[:50]:
                    html_content += f"<li>≠ {f}</li>"
                if len(files) > 50:
                    html_content += f"<li><strong>... e mais {len(files) - 50}
arquivos</strong></li>"
                html_content += '</ul></div>'
            html_content += '</div>'

        html_content += f"""
        </div>
        <div class="footer">
            Gerado em {datetime.now().strftime('%d/%m/%Y às %H:%M:%S')}<br>
            File Integrity Monitor - Monitoramento de Múltiplos Diretórios
        </div>
    </div>
</body>
</html>"""

        with open(report_file, 'w', encoding='utf-8') as f:
            f.write(html_content)

        print(f"Relatório HTML salvo em: {report_file}")
        return report_file

    def send_email_notification(self, comparison, html_file, config):
        """Envia notificação por email"""
        try:
            total = comparison['total_current']
            intact = len(comparison['intact_files'])
```

```
            modified = len(comparison['modified_files'])
            corrupted = len(comparison['corrupted_files'])
            new = len(comparison['new_files'])
            removed = len(comparison['removed_files'])

            # Calcula porcentagens
            intact_pct = (intact / total * 100) if total > 0 else 0
            modified_pct = (modified / total * 100) if total > 0 else 0
            corrupted_pct = (corrupted / total * 100) if total > 0 else 0

            integrity_score = (intact / (intact + modified + corrupted)) * 100 if (intact +
modified + corrupted) > 0 else 0

            if corrupted > 0 or integrity_score < 90:
                priority = "ALTA"
            elif integrity_score < 95:
                priority = "MÉDIA"
            else:
                priority = "NORMAL"

            msg = MIMEMultipart('alternative')
            msg['Subject'] = f"[{priority}] Integridade {len(comparison['target_dirs'])}
Diretórios - {integrity_score:.1f}%"
            msg['From'] = config['from_email']
            msg['To'] = config['to_email']

            dirs_text = "\n".join(f"  • {d}" for d in comparison['target_dirs'])

            text = f"""
Relatório de Integridade - Múltiplos Diretórios
==============================================

Integridade Geral: {integrity_score:.2f}%
Prioridade: {priority}

Diretórios Monitorados ({len(comparison['target_dirs'])}):
{dirs_text}

Scan atual: {comparison['current_timestamp']}
```

```
Estatísticas:
- Arquivos íntegros: {intact} ({intact_pct:.1f}%)
- Arquivos modificados: {modified} ({modified_pct:.1f}%)
- Arquivos corrompidos: {corrupted} ({corrupted_pct:.1f}%)
- Arquivos novos: {new}
- Arquivos removidos: {removed}

Relatório HTML completo em anexo.
"""

            with open(html_file, 'r', encoding='utf-8') as f:
                html = f.read()

            msg.attach(MIMEText(text, 'plain'))
            msg.attach(MIMEText(html, 'html'))

            with smtplib.SMTP(config['smtp_server'], config['smtp_port']) as server:
                if config.get('use_tls', True):
                    server.starttls()
                if config.get('smtp_user') and config.get('smtp_password'):
                    server.login(config['smtp_user'], config['smtp_password'])
                server.send_message(msg)

            print(f"\n✓ Email enviado com sucesso para {config['to_email']}")
            return True

        except Exception as e:
            print(f"\n✗ Erro ao enviar email: {e}")
            return False

    def save_config(self, directories):
        """Salva configuração de diretórios"""
        config = {
            'directories': [str(d) for d in directories],
            'last_updated': datetime.now().isoformat()
        }
        with open(self.config_file, 'w') as f:
            json.dump(config, f, indent=2)

    def load_config(self):
```

```
        """Carrega configuração"""
        if not self.config_file.exists():
            return None
        with open(self.config_file, 'r') as f:
            return json.load(f)

    def save_email_config(self, smtp_server, smtp_port, from_email, to_email,
                          smtp_user=None, smtp_password=None, use_tls=True):
        """Salva configuração de email"""
        config = {
            'smtp_server': smtp_server,
            'smtp_port': smtp_port,
            'from_email': from_email,
            'to_email': to_email,
            'smtp_user': smtp_user,
            'smtp_password': smtp_password,
            'use_tls': use_tls
        }
        with open(self.email_config_file, 'w') as f:
            json.dump(config, f, indent=2)
        print(f"Configuração de email salva em: {self.email_config_file}")

    def load_email_config(self):
        """Carrega configuração de email"""
        if not self.email_config_file.exists():
            return None
        with open(self.email_config_file, 'r') as f:
            return json.load(f)

def main():
    parser = argparse.ArgumentParser(
        description='File Integrity Monitor - Suporta múltiplos diretórios',
        formatter_class=argparse.RawDescriptionHelpFormatter,
        epilog="""
Exemplos:
  %(prog)s /dados                                  # Um diretório
  %(prog)s /dados /backup /www --send-email        # Múltiplos diretórios
  %(prog)s --add-dir /novo/diretorio               # Adiciona diretório
  %(prog)s --list-dirs                             # Lista diretórios configurados
  %(prog)s --config-email                          # Configura email
```

```
    """
)
parser.add_argument('directories', nargs='*', help='Diretórios a monitorar')
parser.add_argument('--database', default='.file_integrity',
                    help='Diretório para dados (padrão: .file_integrity)')
parser.add_argument('--scan-only', action='store_true',
                    help='Apenas escaneia, não gera relatório')
parser.add_argument('--send-email', action='store_true',
                    help='Envia relatório por email')
parser.add_argument('--config-email', action='store_true',
                    help='Configura email')
parser.add_argument('--add-dir', metavar='DIR',
                    help='Adiciona diretório à configuração')
parser.add_argument('--remove-dir', metavar='DIR',
                    help='Remove diretório da configuração')
parser.add_argument('--list-dirs', action='store_true',
                    help='Lista diretórios configurados')
parser.add_argument('--use-config', action='store_true',
                    help='Usa diretórios da configuração')

args = parser.parse_args()

# Modo configuração de email
if args.config_email:
    print("\n=== Configuração de Email ===\n")
    smtp_server = input("Servidor SMTP (ex: smtp.gmail.com): ")
    smtp_port = int(input("Porta SMTP (ex: 587): "))
    from_email = input("Email remetente: ")
    to_email = input("Email destinatário: ")
    smtp_user = input("Usuário SMTP (ou Enter): ") or None
    smtp_password = input("Senha SMTP (ou Enter): ") or None
    use_tls = input("Usar TLS? (s/n): ").lower() != 'n'

    monitor = FileIntegrityMonitor(["/tmp"], args.database)
    monitor.save_email_config(smtp_server, smtp_port, from_email, to_email,
                              smtp_user, smtp_password, use_tls)
    print("\n✓ Configuração salva!")
    return

# Inicializa monitor temporário para operações de config
```

```
    temp_monitor = FileIntegrityMonitor(["/tmp"], args.database)

    # Lista diretórios
    if args.list_dirs:
        config = temp_monitor.load_config()
        if config and config.get('directories'):
            print("\nDiretórios configurados:")
            for i, d in enumerate(config['directories'], 1):
                print(f"  {i}. {d}")
        else:
            print("\nNenhum diretório configurado ainda.")
        return

    # Adiciona diretório
    if args.add_dir:
        if not os.path.isdir(args.add_dir):
            print(f"Erro: {args.add_dir} não é um diretório válido")
            sys.exit(1)

        config = temp_monitor.load_config() or {'directories': []}
        if args.add_dir not in config['directories']:
            config['directories'].append(args.add_dir)
            temp_monitor.save_config(config['directories'])
            print(f"✓ Diretório adicionado: {args.add_dir}")
        else:
            print(f"⚠  Diretório já configurado: {args.add_dir}")
        return

    # Remove diretório
    if args.remove_dir:
        config = temp_monitor.load_config()
        if config and args.remove_dir in config['directories']:
            config['directories'].remove(args.remove_dir)
            temp_monitor.save_config(config['directories'])
            print(f"✓ Diretório removido: {args.remove_dir}")
        else:
            print(f"⚠  Diretório não encontrado na configuração")
        return

    # Determina quais diretórios usar
```

```
directories = []
if args.use_config:
    config = temp_monitor.load_config()
    if config and config.get('directories'):
        directories = config['directories']
    else:
        print("Erro: Nenhum diretório configurado. Use --add-dir primeiro.")
        sys.exit(1)
elif args.directories:
    directories = args.directories
else:
    config = temp_monitor.load_config()
    if config and config.get('directories'):
        print("Usando diretórios da configuração...")
        directories = config['directories']
    else:
        print("Erro: Especifique diretórios ou use --add-dir para configurar")
        sys.exit(1)

# Valida diretórios
for d in directories:
    if not os.path.isdir(d):
        print(f"Erro: {d} não é um diretório válido")
        sys.exit(1)

# Cria monitor e executa scan
monitor = FileIntegrityMonitor(directories, args.database)
monitor.save_config(directories)

scan_data = monitor.scan_directory()
monitor.save_scan(scan_data)

if not args.scan_only:
    comparison = monitor.compare_scans()
    html_file = monitor.generate_report(comparison)

    if args.send_email and comparison is not None:
        config = monitor.load_email_config()
        if config:
            monitor.send_email_notification(comparison, html_file, config)
```

```
            else:
                print("\n⚠  Configure email com --config-email")

if __name__ == '__main__':
    main()
```

COMO USAR

Método 1: Linha de Comando Direta

```
./file_integrity_monitor_multi_dir.py
/var/archivematica/sharedDirectory/transferSource/dataverse/dataverse/ --send-email
```

Nesse exemplo será monitorado o diretório e gerado o html e enviado via e-mail. Deve ser usado pontualmente.

Método 2: Configuração Persistente (RECOMENDADO)

```
# Adiciona diretórios
./file_integrity_monitor_multi_dir.py --add-dir
/var/archivematica/sharedDirectory/transferSource/dataverse/dataverse
./file_integrity_monitor_multi_dir.py --add-dir
/var/archivematica/sharedDirectory/www/AIPsStore
./file_integrity_monitor_multi_dir.py --add-dir
/var/archivematica/sharedDirectory/www/DIPsStore
./file_integrity_monitor_multi_dir.py --add-dir
/var/archivematica/sharedDirectory/AIPbackup

# Lista diretórios
./file_integrity_monitor_multi_dir.py --list-dirs

# Executa (usa os diretórios configurados automaticamente)
./file_integrity_monitor_multi_dir.py --send-email

# Remove diretório
./file_integrity_monitor_multi_dir.py --remove-dir
/var/archivematica/sharedDirectory/AIPbackup
```

** Ideal para automação com cron Não precisa repetir diretórios toda vez**

Configuração no crontab

```
0 2 * * 1 /usr/bin/python3 /opt/scripts-preservacao/checar-
integridade/file_integrity_monitor_multi_dir.py --send-email >>
/var/log/file_integrity.log 2>&1
```

Revision #5
Created 24 October 2025 13:43:11 by Rondineli G. Saad
Updated 29 October 2025 17:26:23 by Rondineli G. Saad