Disk-Folder-File Analyzer (DFA)

A comprehensive command-line tool for analyzing file types and sizes in directories. DFA recursively scans directories, categorizes files by extension, and provides detailed statistics including file counts, total sizes, and identifies largest/smallest files per category.

Features

- **Recursive Directory Scanning**: Analyzes all files in a directory and its subdirectories
- **File Type Analysis**: Groups files by extension and provides comprehensive statistics
- **Configurable Filtering**: Option to include/exclude hidden files and filter by specific extensions
- **Multiple Sorting Options**: Sort results by size, file count, or extension name
- **Human-Readable Output**: File sizes displayed in KB, MB, GB, etc.
- **Tabulated Results**: Clean, formatted table output with headers
- **Logging Support**: Comprehensive logging with configurable levels
- **Export Capabilities**: Save analysis results to text files
- **Robust Error Handling**: Graceful handling of permission errors, interruptions, and invalid paths
- **Input Sanitization**: Secure path validation and sanitization

Installation

1. Ensure you have Python 3.7+ installed
2. Clone or download the DFA files to a directory
3. Make the main script executable:

```
chmod +x main.py
```

Usage

Basic Usage

```
# Analyze current directory using default configuration
python3 main.py

# Analyze specific directory
python3 main.py /path/to/directory

# Use custom configuration file
python3 main.py -c custom_config.json
```

Advanced Usage

```
# Sort by file count and show only top 20 extensions
python3 main.py -s count -m 20

# Save output to file
python3 main.py -o analysis_results.txt

# Skip summary and show only extension table
python3 main.py --no-summary

# Include hidden files
python3 main.py --show-hidden

# Use extension filtering from config
python3 main.py --extension-filter

# Raw byte sizes instead of human-readable
python3 main.py --raw-sizes

# Verbose logging
python3 main.py -v

# Debug mode
python3 main.py --debug
```

Configuration

DFA uses a JSON configuration file (`config.json`) with the following options:

```
{
  "starting_directory": "/home/user",
  "extension_list": [".txt", ".pdf", ".doc", ".docx", ".jpg", ".png", ".mp4", ".mp3"],
  "use_extension_list": false,
  "exclude_hidden_files": true,
  "human_readable_sizes": true,
  "log_level": "INFO",
  "log_file": "dfa.log"
}
```

Configuration Options

- **starting_directory**: Default directory to analyze when none specified
- **extension_list**: List of file extensions to filter by
- **use_extension_list**: Whether to use the extension filter
- **exclude_hidden_files**: Whether to exclude hidden files and directories
- **human_readable_sizes**: Display sizes in KB/MB/GB instead of raw bytes
- **log_level**: Logging level (DEBUG, INFO, WARNING, ERROR, CRITICAL)

- **log_file**: Path to log file

Output Example

```
============================================================
DISK-FOLDER-FILE ANALYZER SUMMARY
============================================================
Scanned Path: /home/user/projects

Total Files: 1,247
Total Size: 2.3 GB
Unique Extensions: 23

Largest File Overall:
  Name: large_video.mp4
  Size: 1.2 GB
  Path: /home/user/projects/media/large_video.mp4

Smallest File Overall:
  Name: empty.txt
  Size: 0 B
  Path: /home/user/projects/docs/empty.txt

EXTENSION ANALYSIS
============================================================

Extension | File Count | Total Size | Largest File        | Largest Size | Smallest File
| Smallest Size
----------|------------|------------|---------------------|--------------|------------------
--|--------------
.mp4      | 15         | 1.8 GB     | large_video.mp4     | 1.2 GB       | small_clip.mp4
  | 2.1 MB
.pdf      | 87         | 156.7 MB   | manual.pdf          | 45.2 MB      | receipt.pdf
| 23.4 KB
.jpg      | 234        | 98.4 MB    | high_res_photo.jpg  | 12.3 MB      | thumbnail.jpg
| 2.1 KB
.txt      | 156        | 2.1 MB     | log_file.txt        | 234.5 KB     | empty.txt
| 0 B
```

Command Line Options

```
positional arguments:
  directory             Directory to analyze (uses config default if not specified)

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Configuration file path (default: config.json)
  -s {size,count,extension}, --sort {size,count,extension}
                        Sort extensions by: size, count, or extension name (default: size)
  -m MAX_EXTENSIONS, --max-extensions MAX_EXTENSIONS
                        Maximum number of extensions to display (default: show all)
```

```
  -o OUTPUT, --output OUTPUT
                        Output file to save results
  --no-summary          Skip summary section in output
  --extension-filter    Use extension filter from configuration (overrides config setting)
  --show-hidden         Include hidden files and directories (overrides config setting)
  --raw-sizes           Display file sizes in raw bytes instead of human-readable format
  -v, --verbose         Enable verbose output (INFO level logging to console)
  --debug               Enable debug output (DEBUG level logging)
  --log-file LOG_FILE   Custom log file path (overrides config setting)
  --version             show program's version number and exit
```

File Structure

```
dfa/
├── main.py           # Main CLI interface
├── config.py         # Configuration management
├── scanner.py        # Directory scanning and file detection
├── stats.py          # Statistics calculation
├── output.py         # Output formatting and display
├── config.json       # Default configuration file
├── dfa.log          # Log file (created when run)
└── README.md        # This documentation
```

Logging

DFA provides comprehensive logging with different levels:

- **DEBUG**: Detailed information for debugging
- **INFO**: General information about program execution
- **WARNING**: Warning messages (shown on console by default)
- **ERROR**: Error messages
- **CRITICAL**: Critical error messages

Logs are written to both console (warnings and above) and log file (all levels based on configuration).

Error Handling

DFA handles various error conditions gracefully:

- **Permission Errors**: Logs inaccessible files and continues
- **Invalid Paths**: Validates and sanitizes all input paths
- **Interrupted Scans**: Graceful shutdown on Ctrl+C
- **Configuration Errors**: Falls back to defaults for invalid config values
- **File System Errors**: Robust handling of file system issues

Performance Notes

- **Memory Efficient**: Uses generators for large directory scans

- **Interrupt Safe**: Can be safely interrupted with Ctrl+C
- **Progress Logging**: Reports progress every 100 directories and 1000 files
- **Scalable**: Handles directories with thousands of files efficiently

Troubleshooting

Permission Denied Errors

- Ensure you have read permissions for the target directory
- Run with appropriate user privileges if needed
- Check log file for specific permission issues

Large Directory Analysis

- Use verbose mode (-v) to monitor progress
- Consider using extension filtering for faster analysis
- Monitor disk space for log files with large directories

Configuration Issues

- Verify JSON syntax in config file
- Check file paths are accessible
- Review log file for configuration warnings

Development Notes

The application is structured as modular components:

- **config.py**: Handles configuration loading and validation
- **scanner.py**: Performs directory traversal and file detection
- **stats.py**: Calculates statistics and manages data structures
- **output.py**: Formats and displays results
- **main.py**: Orchestrates the application and provides CLI interface

Each module can be tested independently and includes test functionality when run directly.

License

This project is provided as-is for educational and personal use.