Arquivador Recursivo de E-mails e Arquivos (`arquiva_subpastas.py`)

1. Objetivo

Este script Python, com interface gráfica para seleção de pasta, tem como objetivo principal organizar arquivos (com foco especial em e-mails `.eml`) que estão dispersos dentro de uma pasta selecionada **e suas subpastas**. A organização é feita **in-place**, ou seja, os arquivos são movidos para subpastas estruturadas por ano e mês (`YYYY/YYYY-MM`) dentro da própria pasta raiz selecionada, transformando uma estrutura potencialmente desorganizada em um arquivo cronológico.

2. Funcionalidades Principais

- **Interface Gráfica para Seleção:** Utiliza Tkinter para permitir ao usuário selecionar facilmente a pasta raiz que contém os arquivos e subpastas a serem organizados.
- **Processamento Recursivo:** Varre a pasta selecionada e **todas as suas subpastas** em busca de arquivos para organizar.
- **Organização *In-Place*:** Move os arquivos para as pastas `YYYY/YYYY-MM` corretas **dentro** da estrutura da pasta raiz selecionada. Não cria uma pasta de arquivamento separada.
- **Exclusão de Pastas:** Ignora automaticamente pastas chamadas `ERROS` e `anos anteriores` (case-insensitive) durante a varredura recursiva, além da própria pasta de logs.
- **Ignora Arquivos Específicos:** Arquivos `.ffs_db` (usados pelo FreeFileSync) são ignorados.
- **Processamento Baseado na Data:**
  - **Arquivos `.eml`:** Extrai a data do cabeçalho `Date`. Tenta múltiplos formatos e encodings (UTF-8, Latin-1). Se a análise falhar, usa a data/hora atual e registra um erro.
  - **Outros Arquivos:** Utiliza a data e hora da última modificação do arquivo (`os.path.getmtime`). Se não for possível obter a data, registra um erro e **não** move o arquivo.
- **Criação Automática de Subpastas:** Cria as subpastas de destino no formato `YYYY/YYYY-MM` (ex: `2024/2024-05`) dentro da pasta raiz selecionada, caso ainda não existam. Incrementa um contador de pastas criadas.
- **Sanitização Inteligente de Nomes:**
  - Remove o prefixo `msg` (ignorando maiúsculas/minúsculas).
  - Substitui caracteres inválidos (`<`, `>`, `:`, `"`, `/`, `\`, `|`, `?`, `*`) por underscores (`_`).
  - Remove caracteres de controle invisíveis.
  - Remove espaços/pontos no início/fim.
  - Normaliza números no início (ex: `001_` vira `1_`).
  - Garante nome não vazio (fallback: "arquivo_renomeado").
- **Prevenção de Caminhos Longos:** Trunca nomes de arquivo se o caminho completo de destino exceder um limite seguro (245 caracteres), tentando preservar a extensão e lidando com caracteres multi-byte (UTF-8).
- **Resolução de Conflitos (Duplicados):**
  - Verifica se um arquivo com o mesmo nome já existe no destino usando `os.path.samefile` para evitar mover/renomear desnecessariamente se for o mesmo arquivo.
  - Se for um arquivo diferente com o mesmo nome, tenta adicionar sufixo numérico (`_1`, `_2`, ...).

    - Se ainda houver conflito ou o nome ficar muito longo, tenta adicionar um timestamp detalhado (`_YYYYMMDDHHMMSSffffff`) ao nome original (antes do sufixo numérico) e trunca novamente se necessário.
    - Registra um erro se não conseguir encontrar um nome único.
- **Distinção entre Mover e Renomear:** Identifica se a operação necessária é mover o arquivo para uma pasta `YYYY/YYYY-MM` diferente ou apenas renomeá-lo (sanitizar/truncar/resolver duplicata) dentro da sua pasta `YYYY/YYYY-MM` atual. Contadores separados para cada ação.
- **Registro Detalhado de Erros:** Cria uma subpasta `ERROS` na raiz selecionada. Erros que impedem o processamento/movimentação (falha na leitura, falha ao obter data, falha ao criar pasta, falha ao mover/renomear, conflitos irresolúveis) são registrados em um arquivo `archive_failures_*.log`.
- **Relatório Final:** Exibe uma janela de resumo (que fecha automaticamente após 5 segundos) detalhando:
    - Número de arquivos movidos para a pasta correta.
    - Número de arquivos renomeados no local (sanitização/truncamento/duplicata).
    - Número de novas pastas `YYYY/YYYY-MM` criadas.
    - Número total de erros encontrados.
    - Instruções para verificar os logs em caso de erros.

3. Modo de Usar

1. **Execute o Script:** Certifique-se de ter o Python 3 instalado. Execute o script `arquiva_subpastas.py` (por exemplo, clicando duas vezes nele ou rodando `python arquiva_subpastas.py` no terminal).
2. **Mensagem Inicial:** Uma pequena janela aparecerá instruindo você a selecionar a pasta principal onde seus arquivos e subpastas estão localizados e serão organizados. Clique em "OK".
3. **Selecione a Pasta Raiz:** Uma janela de diálogo do sistema operacional será aberta. Navegue até a pasta principal que você deseja organizar (o script processará esta pasta e todas as suas subpastas) e clique em "Selecionar pasta" (ou o botão equivalente).
4. **Confirmação:** Outra janela informativa aparecerá, confirmando a pasta selecionada e informando onde os logs de erro serão salvos (na subpasta `ERROS` dentro da pasta selecionada). Clique em "OK" para iniciar o processo.
5. **Aguarde o Processamento:** O script começará a varrer a pasta selecionada e todas as suas subpastas. Ele analisará cada arquivo, determinará a data correta, sanitizará o nome e o moverá para a subpasta `YYYY/YYYY-MM` apropriada dentro da estrutura original. Este processo pode levar algum tempo dependendo da quantidade de arquivos e pastas. Nenhuma janela de progresso é exibida.
6. **Verifique o Resumo Final:** Ao final, uma janela intitulada "Processamento Concluído" aparecerá com um resumo detalhado:
    - Quantos arquivos foram movidos para pastas de data corretas.
    - Quantos arquivos foram apenas renomeados dentro de suas pastas atuais.
    - Quantas novas pastas de data (`YYYY/YYYY-MM`) foram criadas.
    - Quantos erros ocorreram durante o processo.
    - Se houver erros, indicará para verificar a pasta `ERROS`.
    - Esta janela desaparecerá sozinha após 5 segundos, ou você pode clicar em "Fechar Agora".
7. **Consulte os Logs (se necessário):** Se a mensagem final indicou erros, navegue até a pasta raiz que você selecionou, abra a subpasta `ERROS` e procure pelo arquivo `.log` mais recente (ex: `archive_failures_20240521153000.log`). Abra este arquivo em um editor de texto para ver os detalhes dos arquivos que falharam e o motivo.

8. **Verifique a Organização:** Explore a pasta raiz que você selecionou. Os arquivos agora devem estar organizados dentro das subpastas `YYYY/YYYY-MM`. Pastas que ficaram vazias após a movimentação **não** são removidas automaticamente.

4. Especificações Técnicas

- **Linguagem:** Python 3.x
- **Interface Gráfica (GUI):** Tkinter (módulo padrão) para seleção de pasta e mensagens informativas/resumo.
- **Dependências:** Utiliza apenas módulos padrão do Python: `os`, `shutil`, `email`, `logging`, `re`, `datetime`, `tkinter`.
- **Escopo:** Processa a pasta selecionada e todas as suas subpastas recursivamente.
- **Organização:** *In-place* (move arquivos para `YYYY/YYYY-MM` dentro da pasta raiz selecionada).
- **Itens Ignorados:** Pastas `ERROS`, `anos anteriores` (case-insensitive); arquivos `.ffs_db`.
- **Fonte da Data:** Cabeçalho `Date` para `.eml` (com fallback para data atual + erro logado); Data de modificação (`os.path.getmtime`) para outros arquivos (erro logado se falhar).
- **Codificação `.eml`:** Tenta UTF-8, depois Latin-1.
- **Limite de Caminho:** Tenta manter o caminho completo abaixo de 245 caracteres (`MAX_PATH_LENGTH - SAFE_FILENAME_MARGIN`) através de truncamento inteligente (UTF-8 aware).
- **Logging:** Registra apenas `ERROR` level no arquivo `ERROS/archive_failures_YYYYMMDDHHMMSS.log`.
- **Saída:** Estrutura de pastas organizada, arquivo de log (se houver erros), janela de resumo.