File Processing Infrastructure

Overview

TermAI now supports attaching files to prompts! This enables you to:
- Analyze images with vision-capable AI models
- Extract and analyze text from PDF documents
- Include code files for review and explanation
- Attach markdown and text files for summarization

Supported File Types

Images (Vision Support)

- `.jpg`, `.jpeg` - JPEG images
- `.png` - PNG images
- `.gif` - GIF images
- `.webp` - WebP images

Images are automatically encoded as base64 data URLs and sent to vision-capable models like GPT-4 Vision, Claude 3, etc.

PDF Documents

- `.pdf` - PDF files

Text content is extracted from PDFs using the `github.com/ledongthuc/pdf` library. Multi-page PDFs are fully supported.

Text Files

- `.txt` - Plain text files
- `.md`, `.markdown` - Markdown files

Code Files

- `.go` - Go source code
- `.py` - Python
- `.js`, `.ts`, `.jsx`, `.tsx` - JavaScript/TypeScript
- `.java` - Java
- `.c`, `.cpp`, `.cc`, `.h`, `.hpp` - C/C++
- `.rs` - Rust
- `.rb` - Ruby
- `.php` - PHP
- `.sh`, `.bash` - Shell scripts
- `.yaml`, `.yml` - YAML
- `.json` - JSON
- `.xml` - XML
- `.html`, `.css` - Web files
- `.sql` - SQL

Usage

Command-Line Flag

Use the `-file` or `-f` flag to attach files to your prompt:

```
# Single file
termai -file document.pdf "Summarize this document"

# Multiple files
termai -file report.md -file chart.png "Analyze these files"

# Short form
termai -f code.go -f test.go "Review this code"
```

How It Works

1. **File Processing**: When you attach files, TermAI processes them based on their type:
   - **Images**: Encoded as base64 and sent as image content to vision models
   - **PDFs**: Text is extracted and appended to your prompt
   - **Text/Code**: Content is read and appended to your prompt

2. **Message Format**:
   - For images: Uses OpenAI's vision API format with content arrays
   - For text content: Appended to the prompt with clear delimiters

3. **Error Handling**:
   - Files are validated before processing
   - Unsupported file types are rejected with clear error messages
   - Missing or unreadable files are reported

Examples

Analyze an Image

```
termai -f screenshot.png "What's in this image?"
```

Summarize a PDF

```
termai -f research_paper.pdf "Provide a concise summary of the key findings"
```

Code Review

```
termai -f main.go -f utils.go "Review this code for potential issues"
```

Mixed Content

```
termai -f diagram.png -f notes.md "Explain the diagram using the notes for context"
```

Implementation Details

File Processor Module (`internal/fileprocessor/`)

The `fileprocessor` package provides:

```go
type FileAttachment struct {
    Path     string // Original file path
    Type     string // "image", "text", "pdf"
    Content  string // base64 for images, text for others
    MimeType string // MIME type for images
    Name     string // filename
}

func ProcessFile(path string) (*FileAttachment, error)
func ProcessFiles(paths []string) ([]*FileAttachment, error)
```

Provider Interface Updates

The `Message` struct now supports images:

```go
type Message struct {
    Role    string   `json:"role"`
    Content string   `json:"content"`
    Images  []string `json:"images,omitempty"` // base64 data URLs
}
```

OpenAI-Compatible Provider

Messages with images are formatted using the content array structure:

```json
{
  "role": "user",
  "content": [
    {"type": "text", "text": "What's in this image?"},
    {"type": "image_url", "image_url": {"url": "data:image/jpeg;base64,..."}}
  ]
}
```

Limitations & Notes

1. **Model Support**: Vision features require models that support image input (e.g., GPT-4 Vision, Claude 3)
2. **File Size**: Large files may take longer to process and send
3. **PDF Extraction**: Complex PDFs with special formatting may have text extraction limitations
4. **Chat Mode**: Currently, file attachments are only supported in inline mode (not in `termai chat`)

Future Enhancements

Planned improvements:
- [ ] File attachment support in interactive chat mode
- [ ] Image resizing/compression for large images
- [ ] Support for additional file formats (Word, Excel, etc.)

- [ ] Directory context processing
- [ ] File caching to avoid reprocessing

Dependencies

New dependency added:
- `github.com/ledongthuc/pdf` - PDF text extraction

Existing Go standard library packages used:
- `encoding/base64` - Image encoding
- `mime` - MIME type detection
- `os`, `path/filepath` - File operations

Testing

Build and verify:

```
cd /home/ubuntu/termai
go build -o termai
./termai --help
```

The help output should show the new `-file` flag examples.

Status: ✅ Implemented and tested
Build: ✅ Successful compilation
Version Control: ✅ Committed to git