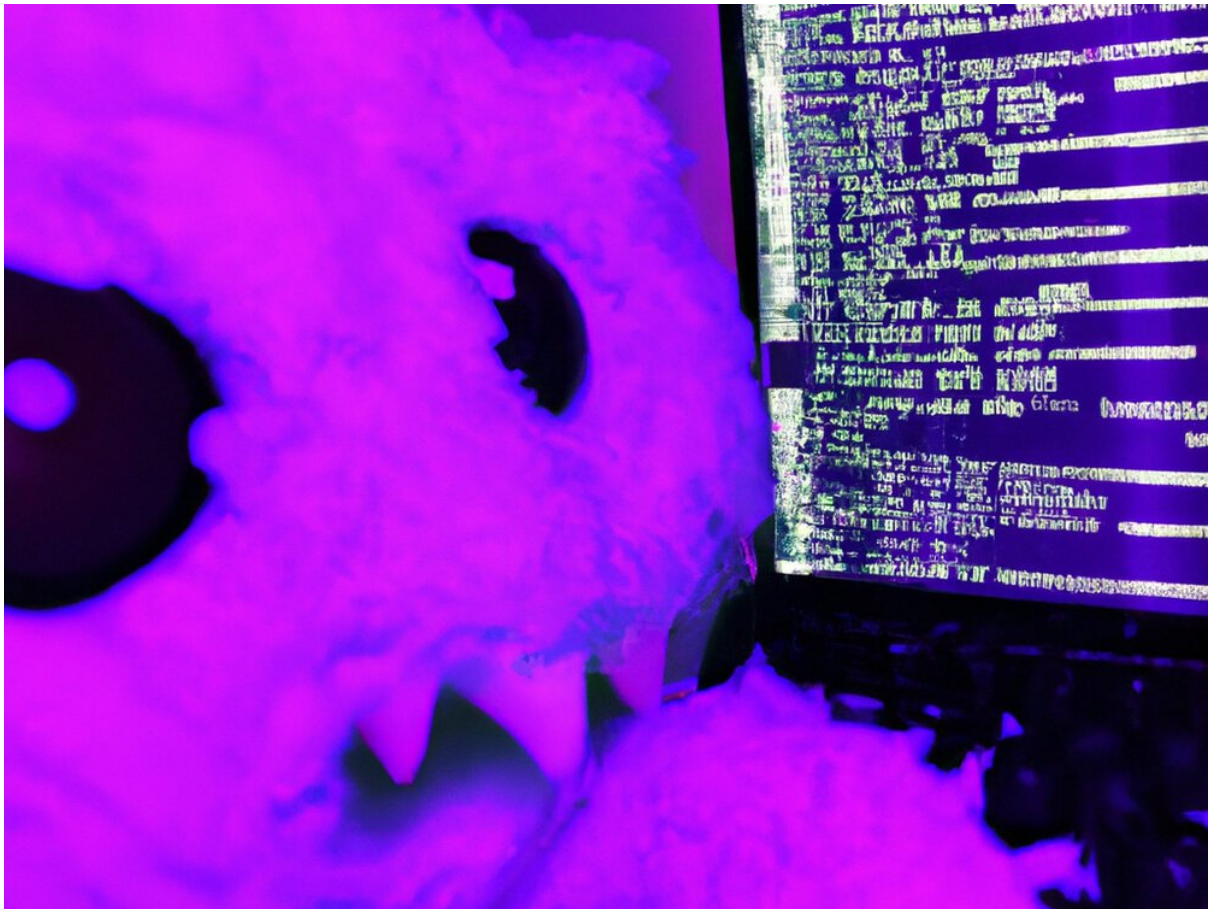
Using AI to generate code

Programming with super powers

Willem L. Middelkoop
May 3, 2023



As part of my software optimisation efforts to cut cloud costs, I needed to replace an existing piece of inefficient server software with something that uses more robust (yet fragmented) tooling available in Debian GNU/Linux. Could the GPT4 language model deliver me some AI magic? Read along!

Part of my company's cloud services is the so-called "Lemmid Manager", an integrated content management system that takes raw input (text, videos, images) and generates beautiful, high-performant static websites. Confronted with surging cloud costs, I am rewriting parts of the software to be more efficient: using less computer power to achieve the same results.

Image rotation

You may not realise it, but whenever you utilise your phone to take a photo, it is always stored landscape (or widescreen). When you take a portrait (or upright) photo, your

phone simply adds a little bit of extra information that saves the way you kept the camera - so that when you look back at the photo, it appears correctly oriented. It is faster to write a single bit of (extra) information than to rotate the actual bits and bytes comprising your image.

This is all fine if the photo doesn't leave your phone, but when one would upload the photo to a website - it needs to be optimised for the web. The actual image bits and bytes must then reflect the actual contents of the photo. The image can then be further optimised for loading speeds and different viewports.

Auto image rotation based on EXIF

Modern (smartphone) camera's save the orientation in the so-called EXIF accompanying the actual image data. In order to have the image (bits and bytes) correctly reflect the camera's orientation, one should read the EXIF data and based on the orientation apply a rotation algorithm to the image bits and bytes.

Current implementation

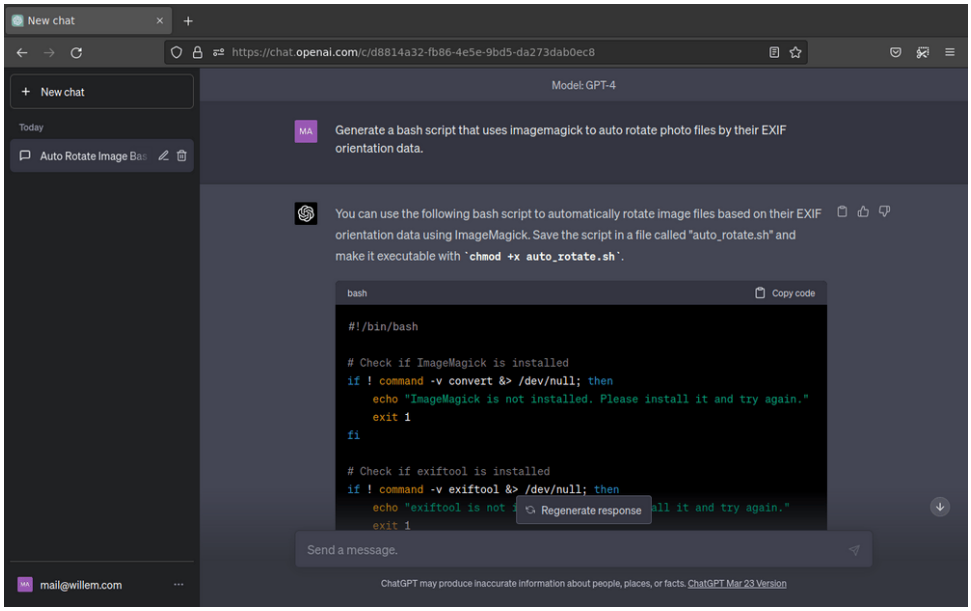
The original (handwritten) implementation uses NodeJS and a module called "lwip" (LightWeight ImageProcessor) to do this. This works fine in most situations, but as images have grown in size (due to more megapixels), the memory requirements have significantly increased. The downside of doing everything in a single NodeJS process is that the entire thing becomes somewhat of a memory hog - causing the complete NodeJS backend to suffer when you push it to its limits processing thousands of photos.

A common strategy to optimise bottlenecks in a piece of software is to extract them from the main program. Instead of processing images from NodeJS directly, I want to use common Debian GNU/Linux tools (that have a proven track record) to do the image auto rotation. From NodeJS I can then call the external 'auto_rotator' and wait for it to complete its task. By having a NodeJS process wait for some asynchronous result/callback, you'll make it available again for other requests.

GPT4 as code generator

The great thing of free (as in libre) software is, that there are many building blocks available for you to use to build your own programs. I knew that Debian GNU/Linux has tools available in that can do the image rotation based on EXIF data. The challenge is to create a single script that uses these different tools to do the job while providing some robustness in handling exceptions.

Instead of deep diving myself into all the available commands and command options, I decided to ask GPT4 to generate a script for me to do precisely what I wanted, utilising standard tools available in Debian GNU/Linux:



Generating code is as easy as typing in a descriptive prompt

Within a few seconds OpenAI's GPT4 generated a bash script that uses ImageMagick (a common graphics tool) to perform modifications based on the EXIF data in a given image file. The AI selected the ”exiftool” to read the orientation data and based on that tool's output, run the ImageMagick's ”convert” command to apply the right rotation.

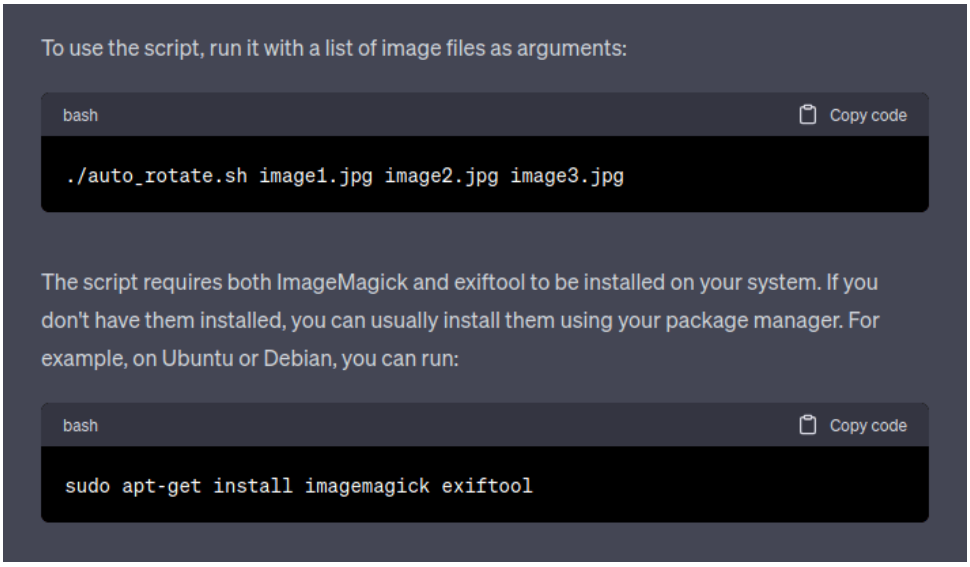
```bash
#!/bin/bash

# Check if ImageMagick is installed
if ! command -v convert &> /dev/null; then
    echo "ImageMagick is not installed. Please install it and try again."
    exit 1
fi

# Check if exiftool is installed
if ! command -v exiftool &> /dev/null; then
    echo "exiftool is not installed. Please install it and try again."
    exit 1
fi

# Check if at least one file is provided
if [ "$#" -eq 0 ]; then
    echo "Usage: ./auto_rotate.sh [file1] [file2] ..."
    exit 1
fi

# Loop through all provided files
for file in "$@"; do
    # Check if the file exists
    if [ ! -f "$file" ]; then
        echo "File '$file' does not exist."
        continue
    fi

    # Extract the orientation using exiftool
    orientation=$(exiftool -Orientation -n -S -s "$file")

    # Perform rotation based on the orientation value
    case "$orientation" in
        1)
            echo "Image '$file' has the correct orientation."
            ;;
        3)
            echo "Rotating image '$file' 180 degrees."
            convert "$file" -rotate 180 "$file"
            ;;
        6)
            echo "Rotating image '$file' 90 degrees clockwise."
            convert "$file" -rotate 90 "$file"
            ;;
        8)
            echo "Rotating image '$file' 90 degrees counterclockwise."
            convert "$file" -rotate -90 "$file"
            ;;
        *)
            echo "Image '$file' has an unsupported orientation value. Skipping."
            ;;
    esac

    # Reset the orientation to normal
    exiftool -Orientation=1 -n -overwrite_original "$file"
done
```

GPT4's generated output code "auto_rotate.sh" rotating images using ImageMagick based on EXIF orientation

The full code (that was generated within seconds) is beautifully commented, giving you a great feeling what's going on. It is structured and optimised for humans to read.

Documentation

In addition to the actual working code, the AI delivered documentation explaining how to use the newly generated software - including installation instructions. This is something that I know most programmers overlook (technical documentation is often an afterthought).



AI's instructions how to use the generated code - most real programmers can learn from its clarity in communication

Conclusion

Within a few minutes I had a working, documented, piece of code that helps me solve a bottleneck in a piece of cloud software. That is simply amazing - it saved me time (that I subsequently used to write this blog post).

One can only wonder what effects large language models like ChatGPT/GPT4 will have on how we work. I fully embrace AI-tools - knowing how to use them will be essential in the near future (I think)!

A programmer from the future - as imagined by DALL-E from OpenAI