```python
from os.path import dirname, join as pjoin
from scipy.io import wavfile
import matplotlib.pyplot as plt
import numpy as np


if __name__ == '__main__':
    plt.style.use('seaborn')
    wave_path = pjoin(dirname(__file__), "voice_test.wav")

    samplerate, data = wavfile.read(wave_path)
    length = data.shape[0] / samplerate

    print(f"Sample rate: {samplerate}")
    print(f"Audio length: {length}")

    # used to normalize the samples
    m = np.amax(np.abs(data))

    # changing from int16 to float32. Note that we need to normalize the
    # amplitudes since int16 has a range from (-32768, +32768) and float32
    # has (-1.0, +1.0)
    data = (data/m).astype(np.float32)

    # voice_test and voice_test_float32 should sound the same
    wavfile.write("voice_test_float.wav", samplerate, data)

    # y will be the output. We create y using data just to have the same shape
    y = np.copy(data)

    alpha = (0.98, 0.5, -0.98, -0.5)
    n_alpha = len(alpha)
    time = np.linspace(0., length, data.shape[0])

    # First window with plots for each alpha
    plt.figure(1)

    for n, a in enumerate(alpha, start=1):
        for i in range(1, data.shape[0]):
            # implementation of forgetful filter
            y[i] = (y[i - 1] * a) + data[i]

        # config name for each file
        wav_name = f"voice_test_alpha{int(a * 100)}.wav"
        if a < 0:
            wav_name = f"voice_test_alpha_neg{abs(int(a * 100))}.wav"
        # saving wav files
        wavfile.write(wav_name, samplerate, y)

        # plotting audio wav for each alpha
        plt.subplot(2, 2, n)
        plt.title(f"Alpha: {a}")
        plt.plot(time, y[:, 0], label="Left channel")
        plt.plot(time, y[:, 1], label="Right channel")
        plt.legend()
        plt.xlabel("Time [s]")
        plt.ylabel("Amplitude")

    # Guarantee space between which subplot
    plt.tight_layout()

    # Second window with original signal
    plt.figure(2)
    plt.title("Original")
    plt.plot(time, data[:, 0], label="Left channel")
    plt.plot(time, data[:, 1], label="Right channel")
    plt.legend()
```

```
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")

plt.show()
```