Inicialmente, deve-se importar o sinal a ser analisado.

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import IPython.display as ipd
from scipy.io import wavfile
import scipy.io
from scipy.fft import irfft, rfft

sinal = '/content/drive/MyDrive/Aulas/Processamento de Digital de Sinais/PROV02/UnknownSound.wav'
fs, wav = wavfile.read(sinal)
ipd.Audio(sinal)
```

Out[1]:

0:00 / 0:05

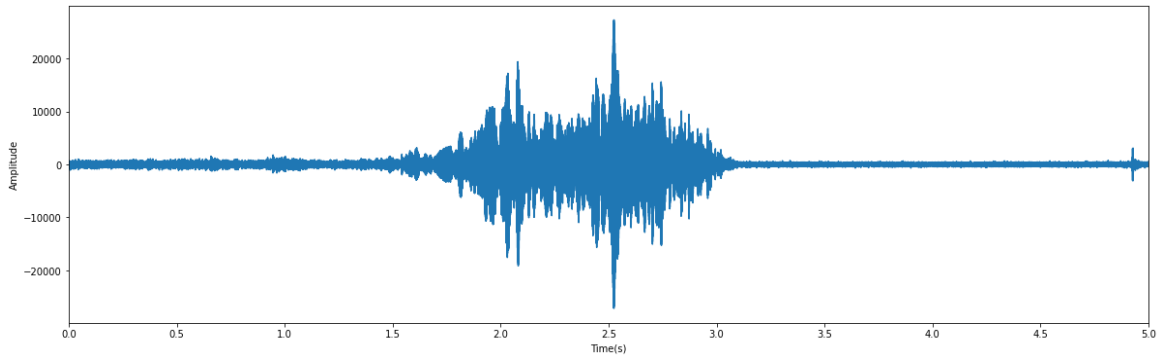
Se é criado um gráfico para visualizar o formato do sinal no domínio do tempo e no domínio da frequência.

In [41]:

```
NT = len(wav)
nt = np.arange(NT)
T = nt/fs

plt.figure(figsize = (20, 6))
plt.plot(T, wav)
plt.xlim(xmin=0,xmax=5)
plt.xticks( np.arange(0,5.5,0.5) )
plt.xlabel('Time(s)')
plt.ylabel('Amplitude')
```

Out[41]:

```
Text(0, 0.5, 'Amplitude')
```



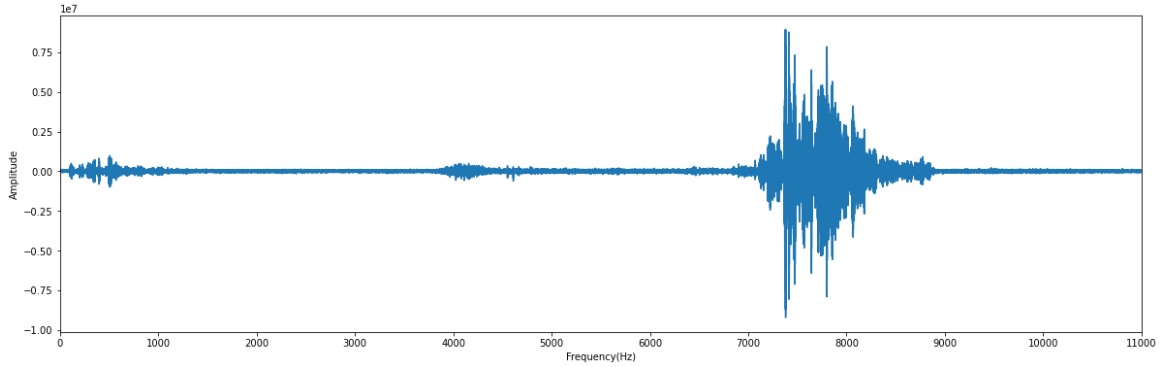
In [53]:

```
from scipy.fft import fft
X = fft(wav)

N = len(X)
n = np.arange(N)
t = N/fs
freq = n/t

plt.figure(figsize = (20, 6))
plt.plot(freq, X)
plt.xlim(xmin=0,xmax=10000)
plt.xticks( np.arange(0,12000,1000) )
plt.xlabel('Frequency(Hz)')
plt.ylabel('Amplitude')
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:131
7: ComplexWarning: Casting complex values to real discards the imaginary
part
  return np.asarray(x, float)
```

Out[53]:

```
Text(0, 0.5, 'Amplitude')
```



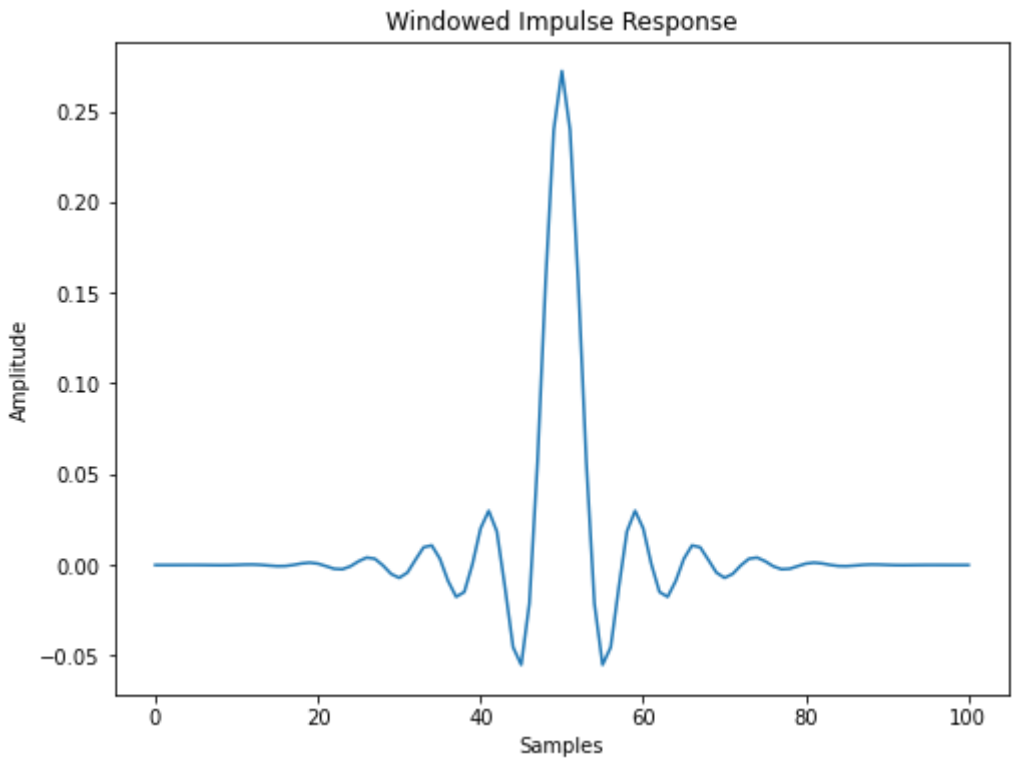
Questão 1- A janela de Kaiser é criada com 101 coeficientes e frequência de corte de 3000 Hz que corresponde à parte inferior do espectograma.

In [83]:

```
import math
from scipy import signal

fc = 3000
wc =  fc/(fs/2)
M = 101
nw = np.arange(M)
r = 100 #Atenuação em DBs
beta = 0.1102*(r- 8.7)

k = scipy.signal.firwin(M, wc, window = ('kaiser',beta))

plt.figure(figsize = (8, 6))
plt.plot(nw, k)
plt.title('Windowed Impulse Response')
plt.xlabel('Samples')
plt.ylabel('Amplitude')
```

Out[83]:

```
Text(0, 0.5, 'Amplitude')
```

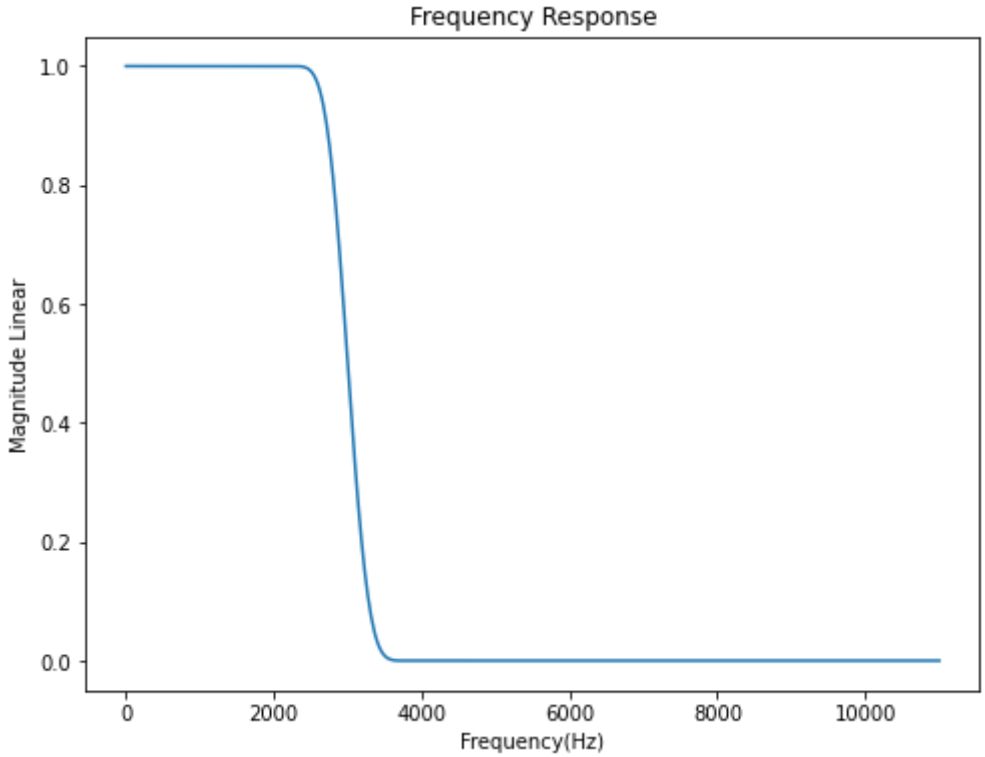


In [84]:

```
K,f = scipy.signal.freqz(k,1,fs = fs)

plt.figure(figsize = (8, 6))
plt.plot(K, abs(f))
plt.title('Frequency Response')
plt.xlabel('Frequency(Hz)')
plt.ylabel('Magnitude Linear')
```

Out[84]:

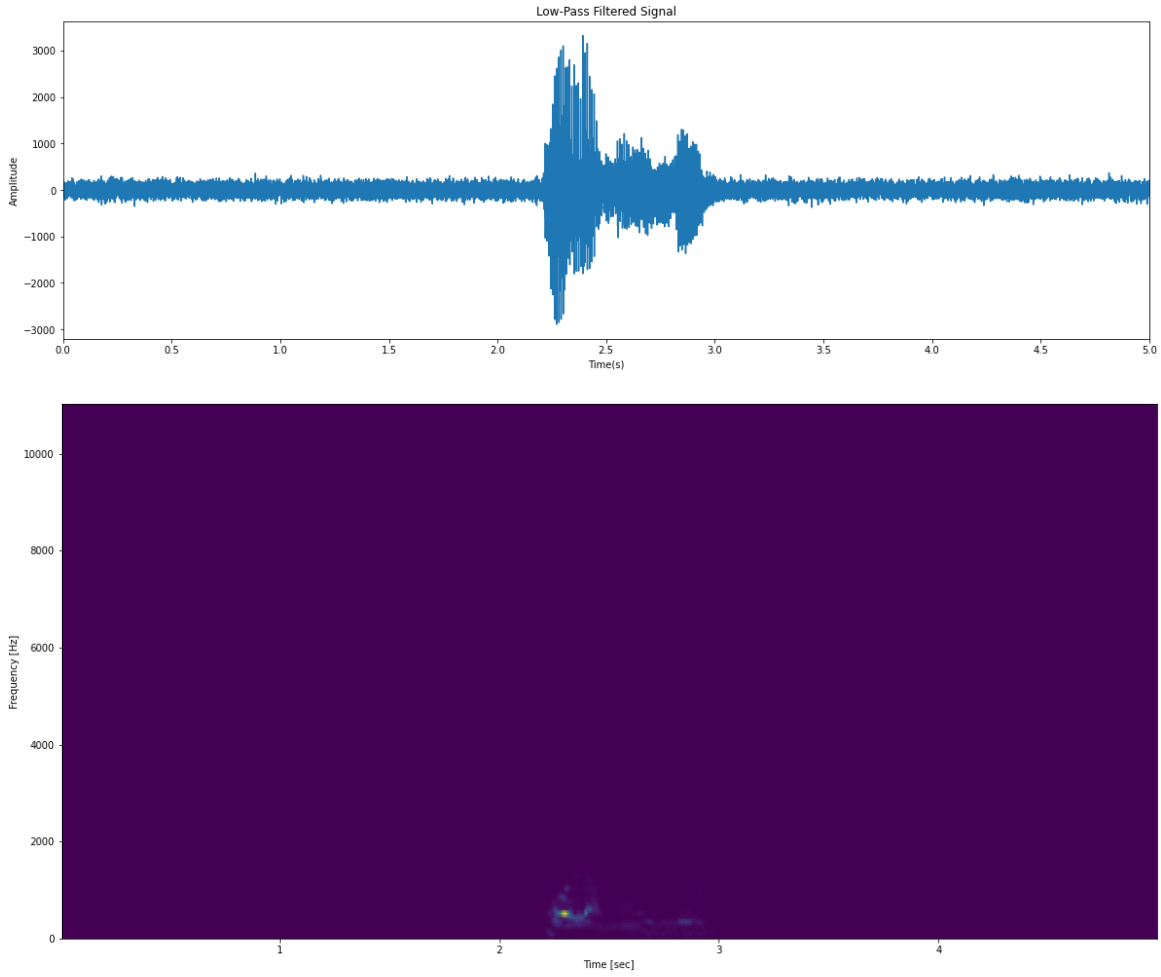
```
Text(0, 0.5, 'Magnitude Linear')
```



Após criar o filtro Low-Pass, o sinal deve ser filtrado, removendo as componentes de alta frequência.

In [90]:

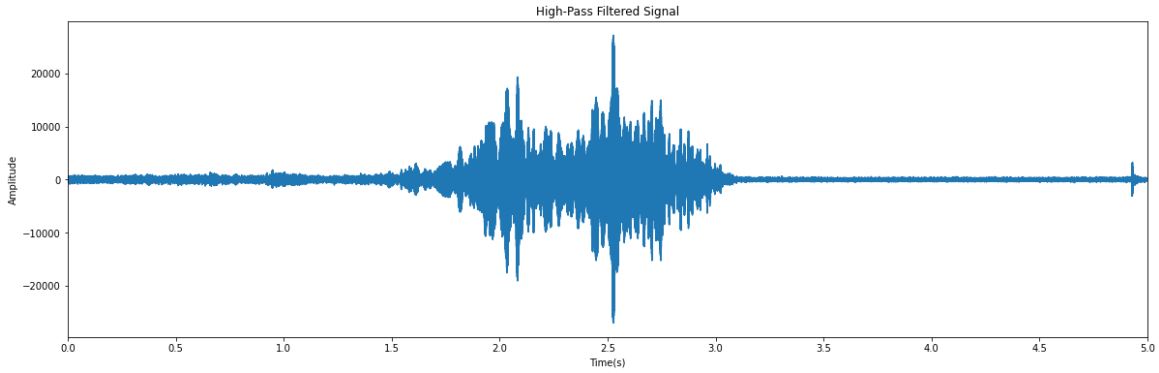
```
sinalLP = scipy.signal.lfilter(k,1,wav)

plt.figure(figsize = (20, 6))
plt.plot(T, sinalLP)
plt.xlim(xmin=0,xmax=5)
plt.xticks( np.arange(0,5.5,0.5) )
plt.title('Low-Pass Filtered Signal')
plt.xlabel('Time(s)')
plt.ylabel('Amplitude')


##############################################################

def plot_spec(Y, fs):
  plt.figure(figsize=(20, 10))
  ff, tt, sxx = scipy.signal.spectrogram(Y,fs,scaling='spectrum')
  plt.pcolormesh(tt, ff, sxx, shading='gouraud')
  plt.ylabel('Frequency [Hz]')
  plt.xlabel('Time [sec]')
  plt.show()

plot_spec(sinalLP,fs)
```

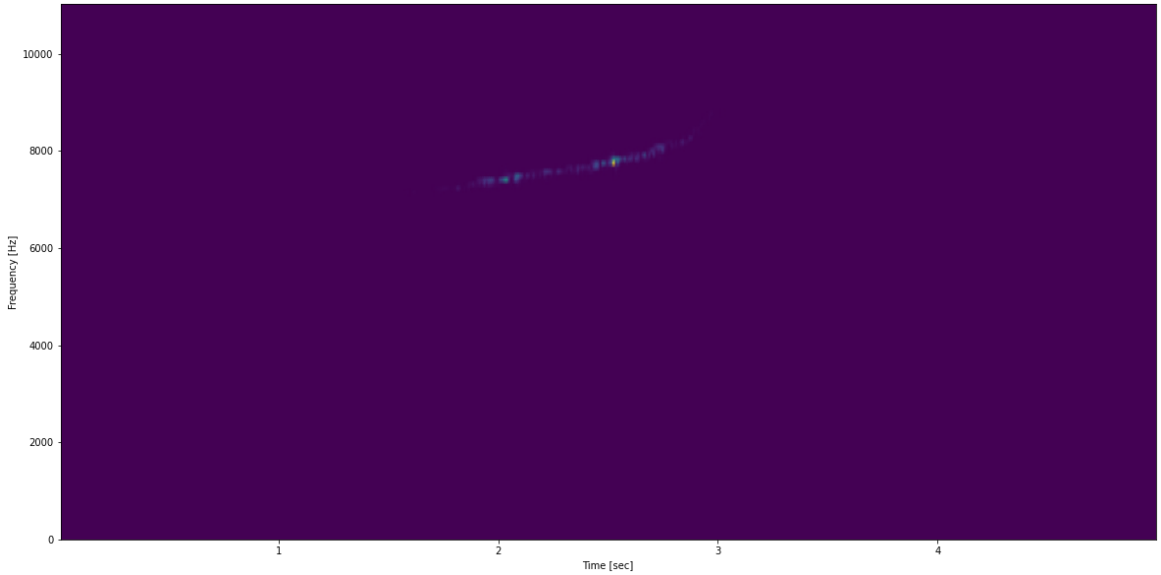


Questão 2- A partir do filtro Low-Pass é possível criar um filtro High-Pass, que mantém as componentes de alta-frequência. Este filtro é aplicado ao sinal para obter apenas a parte superior do espectograma.

In [97]:

```
max_value = np.amax(k)
pos_M = np.where(k == max_value)
b_impulse = np.zeros(M)
b_impulse[pos_M] = 1
b_HP = b_impulse - k

sinalHP = scipy.signal.lfilter(b_HP,1,wav)

plt.figure(figsize = (20, 6))
plt.plot(T, sinalHP)
plt.xlim(xmin=0,xmax=5)
plt.xticks( np.arange(0,5.5,0.5) )
plt.title('High-Pass Filtered Signal')
plt.xlabel('Time(s)')
plt.ylabel('Amplitude')


############################################################

def plot_spec(Y, fs):
  plt.figure(figsize=(20, 10))
  ff, tt, sxx = scipy.signal.spectrogram(Y,fs,scaling='spectrum')
  plt.pcolormesh(tt, ff, sxx, shading='gouraud')
  plt.ylabel('Frequency [Hz]')
  plt.xlabel('Time [sec]')
  plt.show()

plot_spec(sinalHP,fs)
```





Questão 2- b)É possível obter um filtro High-Pass a partir dos coeficientes de um filtro Low-Pass pois os coeficientes High-Pass se comportam de forma semelhante à reflexão dos coeficientes Low-Pass no eixo X. Sendo o coeficiente de maior valor subtraído de 1.