In [216]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
import scipy.stats as stats

def is_float(string):
    try:
        float(string)
        return True
    except ValueError:
        return False

data1 = np.genfromtxt('data\data4_26_24\data_good\p420v1200s3.csv', delim
data2 = np.genfromtxt('data\data4_26_24\data_good\p415v1300s3.csv', delim
data3 = np.genfromtxt('data\data4_26_24\data_good\p417v1400s3.csv', delim

v_data_1 = [float(row[0]) if is_float(row[0]) else np.nan for row in data
i_data_1 = [float(row[1]) if is_float(row[1]) else np.nan for row in data
sigma_i_1 = np.asarray(i_data_1, dtype=np.float64)*0.001
v_data_2 = [float(row[0]) if is_float(row[0]) else np.nan for row in data
i_data_2 = [float(row[1]) if is_float(row[1]) else np.nan for row in data
sigma_i_2 = np.asarray(i_data_1, dtype=np.float64)*0.001
v_data_3 = [float(row[0]) if is_float(row[0]) else np.nan for row in data
i_data_3 = [float(row[1]) if is_float(row[1]) else np.nan for row in data
sigma_i_3 = np.asarray(i_data_1, dtype=np.float64)*0.001

# constants
k = 1.380649E-23
e = 1.602176634E-19

# linear fit function
def func(x, a, b):
    return a*x + b
```

Important Note:

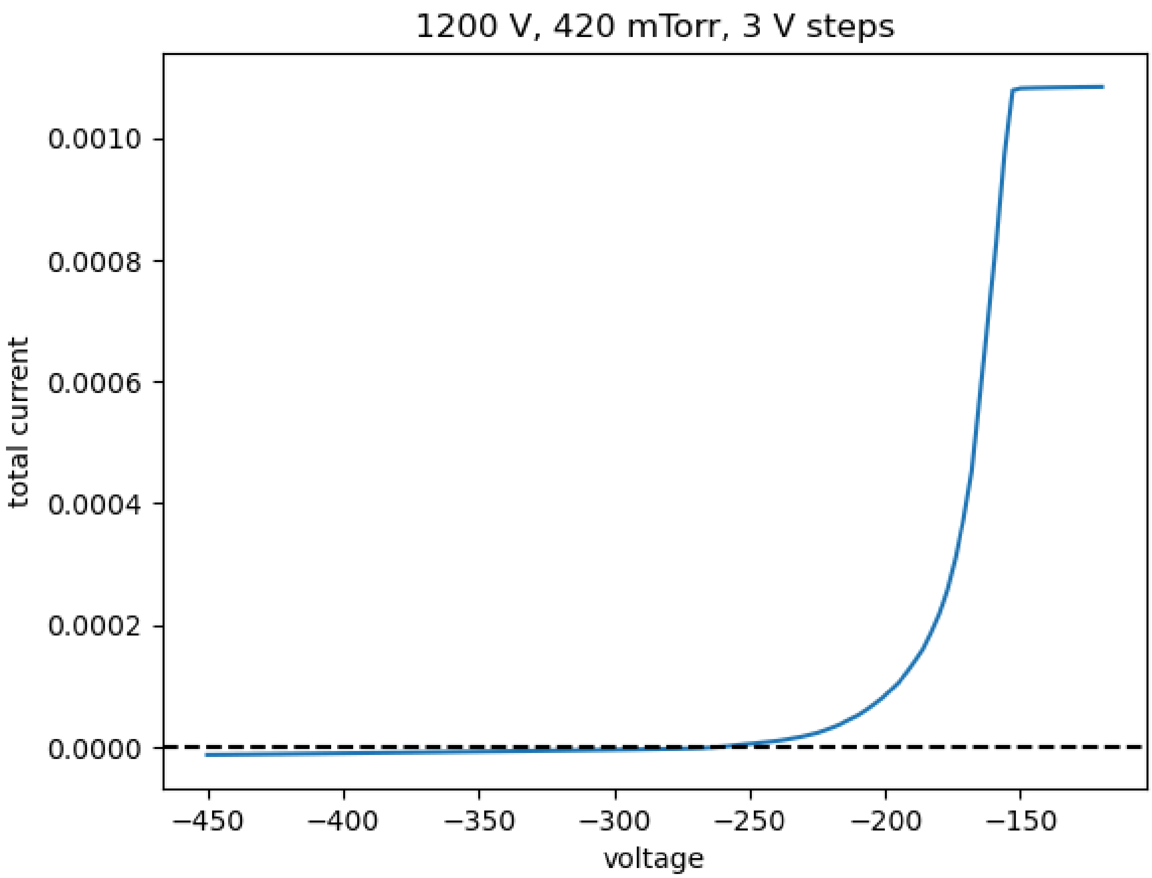
----From what it seems like from this write up (https://download.tek.com/document/LowCurtMsmntsAppNote.pdf (https://download.tek.com/document/LowCurtMsmntsAppNote.pdf)), we can estimate the error in the current to be roughly 0.01 of the current measurement. This has been calculated from our data above.----

I just used the variance from the matrix given to use through pcov in curve_fit to calculate the error. I think sigma_a is essentially the error in our current.

1200 V, 420 mTorr, 3 V steps

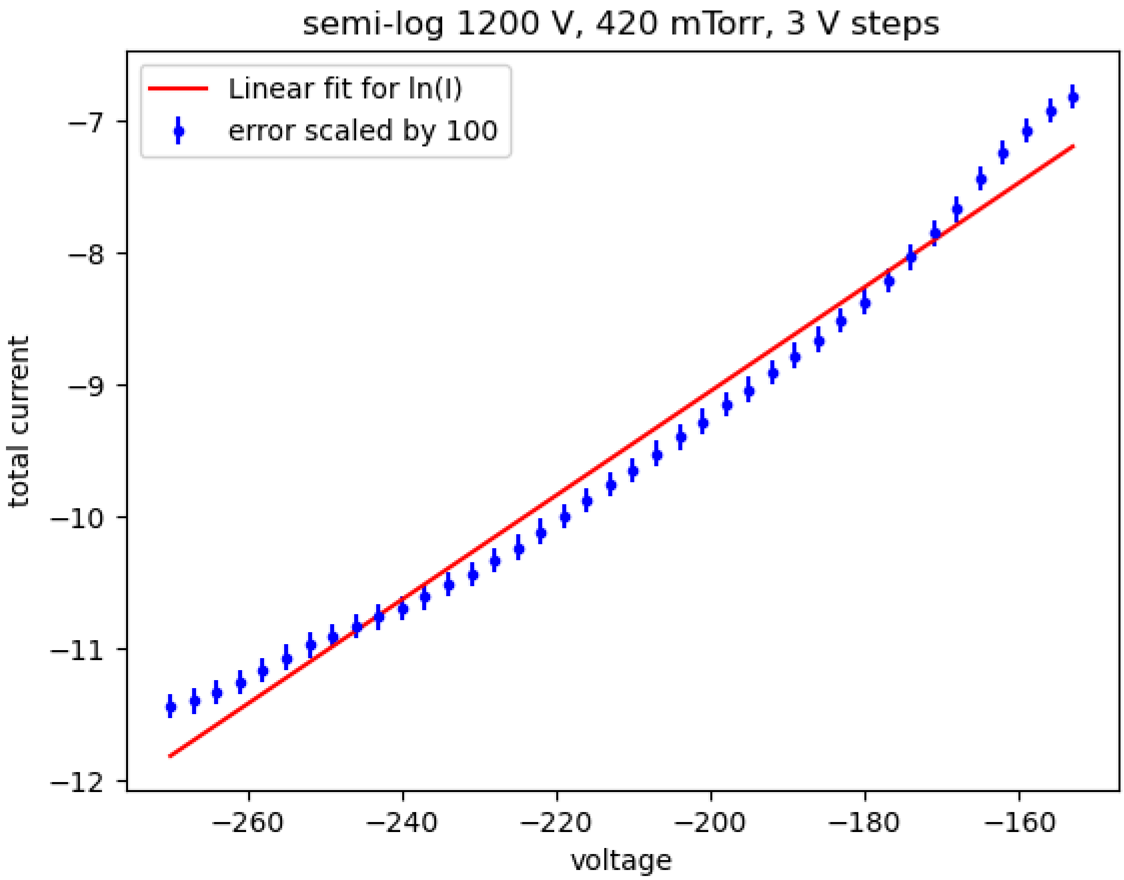
In [253]:

```
# 1200 - initial plot, non-fit

plt.figure()
plt.plot(v_data_1, i_data_1)
plt.axhline(y=0, color='black', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('1200 V, 420 mTorr, 3 V steps')
plt.show()
```



In [254]:
```
I_is_1 = -1.284397e-05 # Ion saturation current for data1

I_data_1 = [x - I_is_1 for x in i_data_1]

#I_data_1
I_data_1_filtered = I_data_1[60:100]
V_data_1 = v_data_1[60:100]
logI_data_1 = np.log(I_data_1_filtered)

popt, pcov = curve_fit(func, V_data_1, logI_data_1, p0=[1.0, 1.0])

V_data_1arr = np.asarray(V_data_1, dtype=np.float64)
logI_fit_1 = func(V_data_1arr, popt[0], popt[1])

a, b = popt
sigma_a, sigma_b = np.sqrt(np.diag(pcov))
fit = f'ln(I) = ({a:.3f} +/- {sigma_a:.3f})* V + ({b:.3f} +/- {sigma_b:.3
print("Equation of the fitted line:", fit)

plt.figure()
plt.errorbar(V_data_1, logI_data_1, yerr=sigma_a*100, fmt='.', color='b',
plt.plot(V_data_1arr, logI_fit_1, color='r', label= 'Linear fit for ln(I)
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('semi-log 1200 V, 420 mTorr, 3 V steps')
plt.legend()
plt.show()

print(f'Electron temperature: {(e/(k*a)):.0f} K +/- {(e/(k*a))*sigma_a:.0
```

```
Equation of the fitted line: ln(I) = (0.039 +/- 0.001)* V + (-1.159 +/- 0.20
1)
```




```
Electron temperature: 294038 K +/- 276 K
```

In [230]:

```
sigma_logI_1 = sigma_i_1[66:100]/I_data_1_filtered
np.size(sigma_logI_1)
np.size(v_data_1)
```

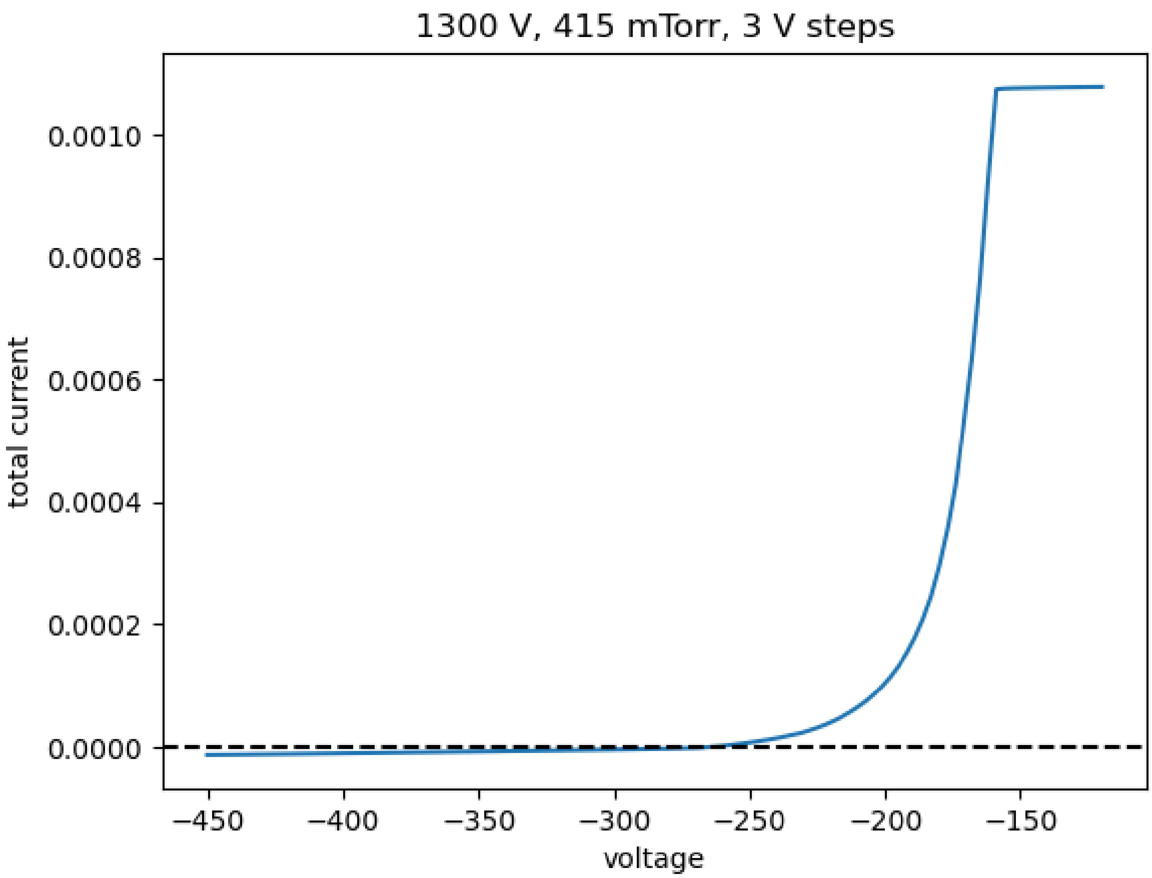
Out[230]:

```
111
```

1300 V, 415 mTorr, 3 V steps

In [256]:

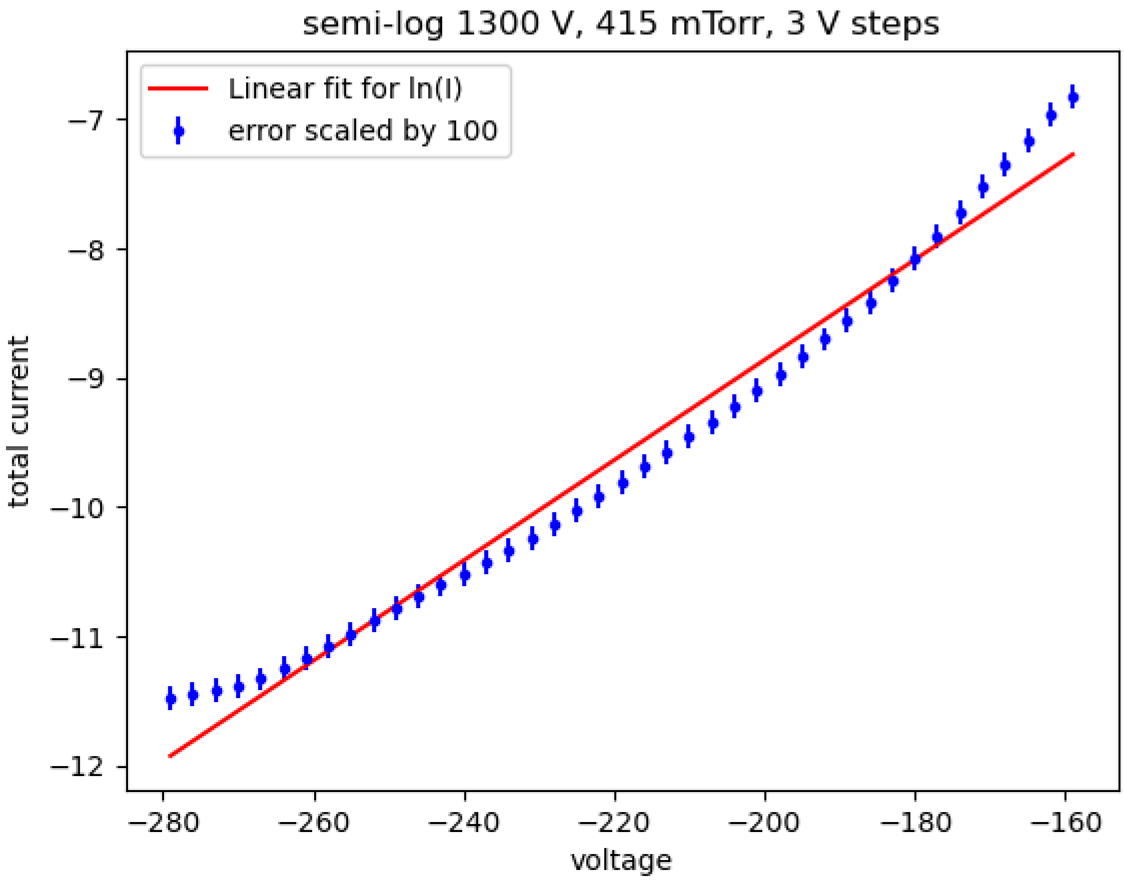
```
# 3200 - initial plot, non-fit

plt.figure()
plt.plot(v_data_2, i_data_2)
plt.axhline(y=0, color='black', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('1300 V, 415 mTorr, 3 V steps')
plt.show()
```



In [263]:

```
I_is_2 = -1.315807e-05
#print(i_data_2)

I_data_2 = [x - I_is_2 for x in i_data_2]

I_data_2_filtered = I_data_2[57:98]
V_data_2 = v_data_2[57:98]
logI_data_2 = np.log(I_data_2_filtered)

popt, pcov = curve_fit(func, V_data_2, logI_data_2, p0=[1.0,1.0])

V_data_2arr = np.asarray(V_data_2, dtype=np.float64)
logI_fit_2 = func(V_data_2arr, popt[0], popt[1])

a, b = popt
sigma_a, sigma_b = np.sqrt(np.diag(pcov))

fit = f'ln(I) = ({a:.3f} +/- {sigma_a:.3f})* V + ({b:.3f} +/- {sigma_b:.3
print("Equation of the fitted line:", fit)

plt.figure()
#plt.plot(V_data_2, logI_data_2)
plt.errorbar(V_data_2, logI_data_2, yerr=sigma_a*100, fmt='.', color='b',
plt.plot(V_data_2arr, logI_fit_2, color='r', label= 'Linear fit for ln(I)
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('semi-log 1300 V, 415 mTorr, 3 V steps')
plt.legend()
plt.show()

print(f'Electron temperature: {(e/(k*a)):.0f} K +/- {(e/(k*a))*sigma_a:.0
```

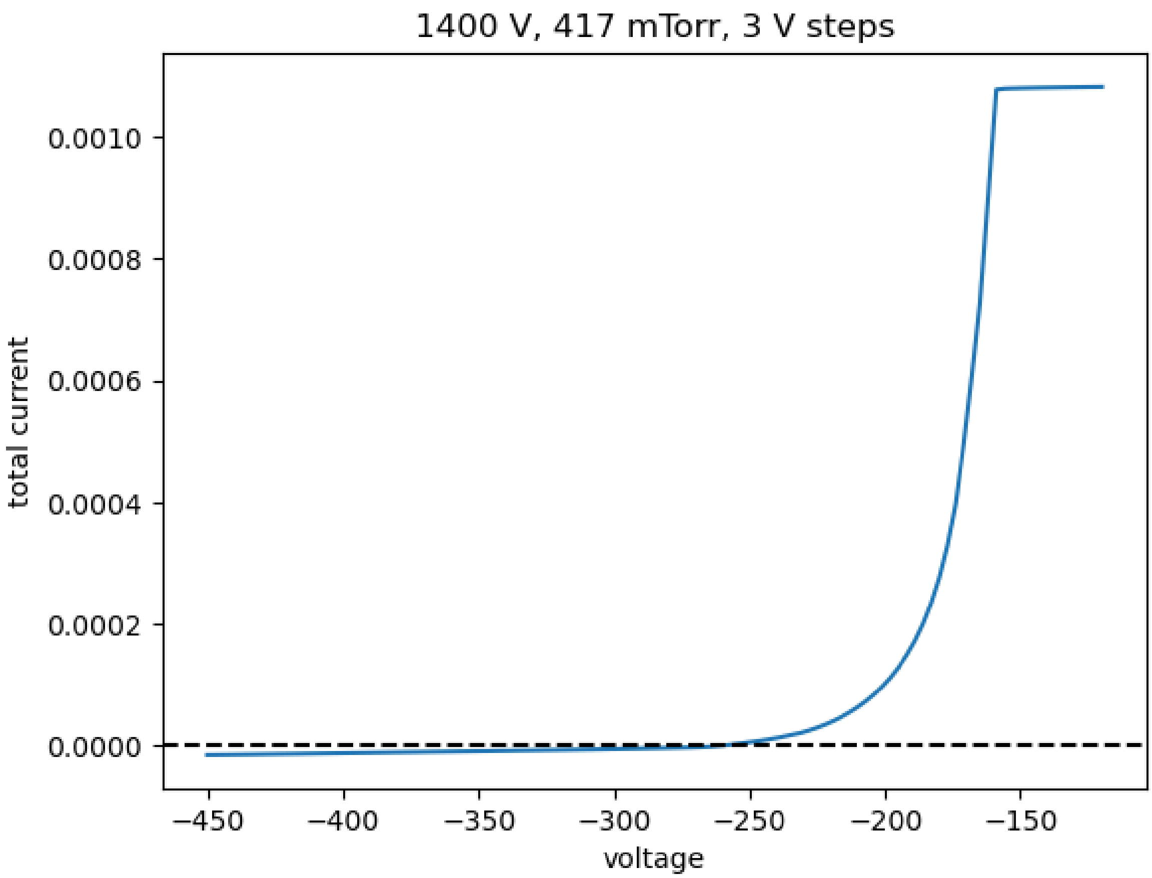
```
Equation of the fitted line: ln(I) = (0.039 +/- 0.001)* V + (-1.111 +/- 0.20
4)
```




```
Electron temperature: 299393 K +/- 276 K
```

1400 V, 417 mTorr, 3 V steps

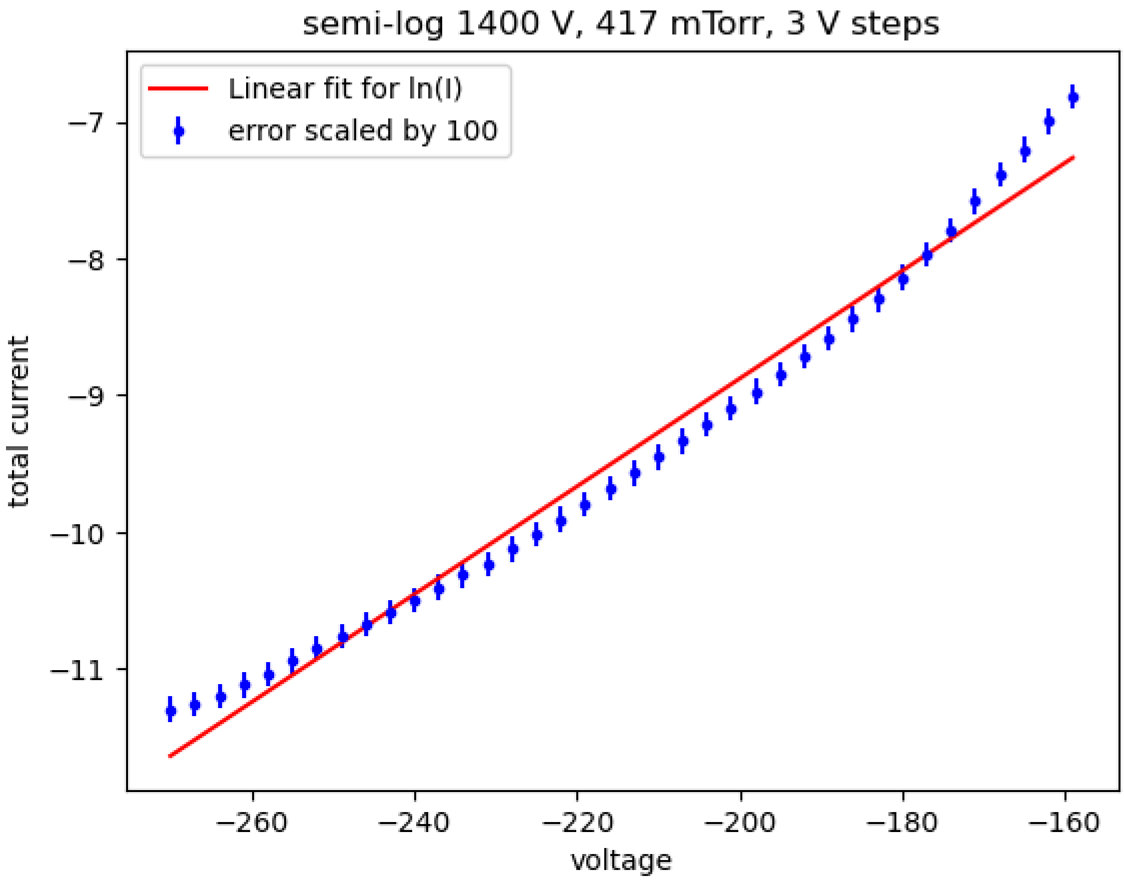
In [179]:
```python
# 1200 - initial plot, non-fit

plt.figure()
plt.plot(v_data_3, i_data_3)
plt.axhline(y=0, color='black', linestyle='--')
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('1400 V, 417 mTorr, 3 V steps')
plt.show()
```



In [264]:

```
I_is_3 = -1.478699e-05

I_data_3 = [x - I_is_3 for x in i_data_3]

I_data_3_filtered = I_data_3[60:98]
V_data_3 = v_data_3[60:98]
logI_data_3 = np.log(I_data_3_filtered)

popt, pcov = curve_fit(func, V_data_3, logI_data_3, p0=[1.0,1.0])

V_data_3arr = np.asarray(V_data_3, dtype=np.float64)
logI_fit_3 = func(V_data_3arr, popt[0], popt[1])

a, b = popt
sigma_a, sigma_b = np.sqrt(np.diag(pcov))
fit = f'ln(I) = ({a:.3f} +/- {sigma_a:.3f})* V + ({b:.3f} +/- {sigma_b:.3
print("Equation of the fitted line:", fit)

plt.figure()
#plt.plot(V_data_3, logI_data_3)
plt.errorbar(V_data_3, logI_data_3, yerr=sigma_a*100, fmt='.', color='b',
plt.plot(V_data_3arr, logI_fit_3, color='r', label= 'Linear fit for ln(I)
plt.xlabel('voltage')
plt.ylabel('total current')
plt.title('semi-log 1400 V, 417 mTorr, 3 V steps')
plt.legend()
plt.show()

print(f'Electron temperature: {(e/(k*a)):.0f} K +/- {(e/(k*a))*sigma_a:.0
```

```
Equation of the fitted line: ln(I) = (0.039 +/- 0.001)* V + (-1.001 +/- 0.19
9)
```




```
Electron temperature: 294587 K +/- 270 K
```

In [ ]:

```
1
```