In [32]:
```
import numpy as np
import download_data as dl
import matplotlib.pyplot as plt
import sklearn.svm as svm
from sklearn import metrics
from conf_matrix import func_confusion_matrix
import pandas as pd
```

In [3]:
```
## step 1: load data from csv file.
data = dl.download_data('crab.csv').values

n = 200
#split data
S = np.random.permutation(n)
#100 training samples
Xtr = data[S[:100], :6]
Ytr = data[S[:100], 6:]
# 100 testing samples
X_test = data[S[100:], :6]
Y_test = data[S[100:], 6:].ravel()
```

In [3]:
```
data
```

Out[3]:
```
array([[ 0. , 20.6, 14.4, ..., 46.5, 19.6, -1. ],
       [ 1. , 13.3, 11.1, ..., 32.3, 11.3, -1. ],
       [ 0. , 16.7, 14.3, ..., 37. , 14.7,  1. ],
       ...,
       [ 1. ,  8.8,  7.7, ..., 20.8,  7.4, -1. ],
       [ 1. , 16.2, 15.2, ..., 40.1, 13.9,  1. ],
       [ 0. , 15.6, 14. , ..., 35.3, 13.8,  1. ]])
```

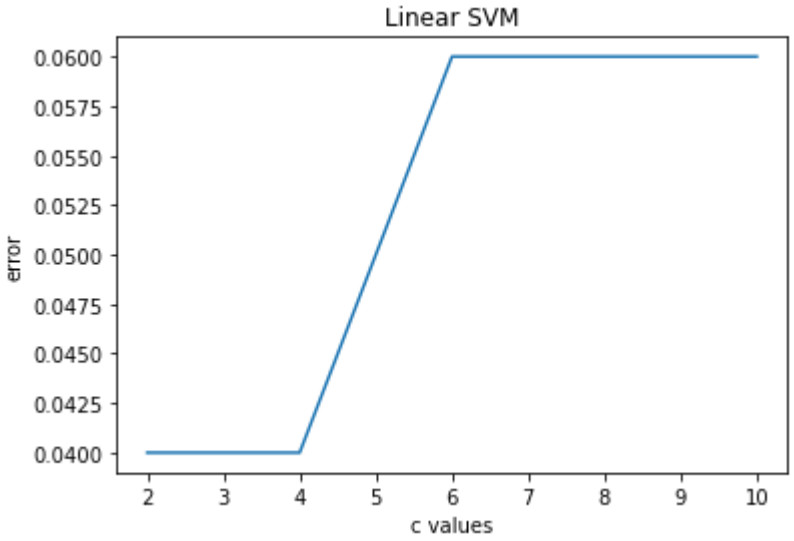
In [22]:
```
## step 2 randomly split Xtr/Ytr into two even subsets: use one for tr
##############placeholder 1: training/validation ######################
n2 = len(Xtr)

#split data
S = np.random.permutation(n2)

# subsets for training models
x_train= Xtr[S[:50], :]
y_train= Ytr[S[:50], :]
# subsets for validation
x_validation= Xtr[S[50:], :]
y_validation= Ytr[S[50:], :]
##############placeholder end ######################
```

In [24]:

```
c_range = [2,4,6,8,10]
svm_c_error = []
for c_value in c_range:
    model = svm.SVC(kernel='linear', C=c_value)
    model.fit(X=x_train, y=y_train.reshape(50,))
    error = 1. - model.score(x_validation, y_validation.reshape(50,))
    svm_c_error.append(error)
plt.plot(c_range, svm_c_error)
plt.title('Linear SVM')
plt.xlabel('c values')
plt.ylabel('error')
#plt.xticks(c_range)
plt.show()
```

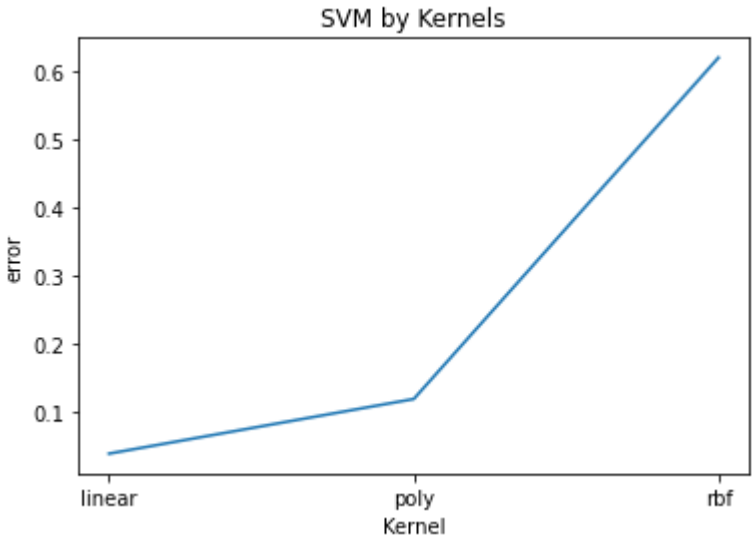


In [25]:

```
kernel_types = ['linear', 'poly', 'rbf']
svm_kernel_error = []
for kernel_value in kernel_types:
    # your own codes
    model = svm.SVC(kernel=kernel_value, C=2)
    model.fit(X=x_train, y=y_train.reshape(50,))
    error = 1. - model.score(x_validation, y_validation.reshape(50,))

    svm_kernel_error.append(error)

plt.plot(kernel_types, svm_kernel_error)
plt.title('SVM by Kernels')
plt.xlabel('Kernel')
plt.ylabel('error')
plt.xticks(kernel_types)
plt.show()
```



In [29]:

```
best_kernel = 'linear'
best_c = 2
# train a SVM model with the best hyper-parameters.
model = svm.SVC(kernel=best_kernel, C=best_c)
model.fit(X=x_train, y=y_train.reshape(50,))
print('Testing Score:',model.score(X_test,Y_test))
print('Testing Error:',1. - model.score(X_test,Y_test))
```

```
Testing Score: 0.94
Testing Error: 0.06000000000000005
```

In [28]:
```
y_pred = model.predict(X_test)
conf_matrix, accuracy, recall_array, precision_array = func_confusion_

print("Confusion Matrix: ")
print(conf_matrix)
print("Average Accuracy: {}".format(accuracy))
print("Per-Class Precision: {}".format(precision_array))
print("Per-Class Recall: {}".format(recall_array))
```

```
Confusion Matrix:
[[42  6]
 [ 0 52]]
Average Accuracy: 0.94
Per-Class Precision: [1.         0.89655172]
Per-Class Recall: [0.875 1.   ]
```

In [34]:
```
d = {'Y':Y_test,
    'Yhat':y_pred}

cf = pd.DataFrame(d)
cf['Error'] = 'NULL'
for i in range(0,len(y_pred)):
    if cf['Yhat'].values[i] > cf['Y'].values[i]:
        cf['Error'].values[i] = 'FALSE POSITVE'
    if cf['Yhat'].values[i] < cf['Y'].values[i]:
        cf['Error'].values[i] = 'FALSE NEGATIVE'
    if cf['Yhat'].values[i] == cf['Y'].values[i]:
        cf['Error'].values[i] = 'NO ERROR'

cf.sort_values(by=['Error'],inplace= True)
cf
```

Out[34]:

	Y	Yhat	Error
37	1.0	-1.0	FALSE NEGATIVE
31	1.0	-1.0	FALSE NEGATIVE
84	1.0	-1.0	FALSE NEGATIVE
14	1.0	-1.0	FALSE NEGATIVE
60	1.0	-1.0	FALSE NEGATIVE
...	...	...	...
28	-1.0	-1.0	NO ERROR
27	1.0	1.0	NO ERROR
26	1.0	1.0	NO ERROR
50	1.0	1.0	NO ERROR
99	-1.0	-1.0	NO ERROR

100 rows × 3 columns

In [35]:
```
1 cf.loc[cf['Error']!= 'NO ERROR']
```

Out[35]:

	Y	Yhat	Error
37	1.0	-1.0	FALSE NEGATIVE
31	1.0	-1.0	FALSE NEGATIVE
84	1.0	-1.0	FALSE NEGATIVE
14	1.0	-1.0	FALSE NEGATIVE
60	1.0	-1.0	FALSE NEGATIVE
73	1.0	-1.0	FALSE NEGATIVE

In [ ]:
```
1
```