Pedram - Mirelmi - 610398176

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from MyFCFS import *
```

base condition

In [ ]:
```
number_of_tests = 500
wished_range = 1 * 10**3
context_switch_time = 0
tester = SimpleFCFS(n = 1 * 10**1,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                    )
result = tester.runTestNTimes(number_of_tests)
printUniformTestResult(number_of_tests, wished_range, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with Uniform distribution with low 0 and high 1000
{
    "response times": {
        "mean of mean": 1851.063,
        "mean of std": 1198.15311735154,
        "std of mean": 490.45152226392366,
        "std of std": 284.5114400525691
    },
    "waiting times": {
        "mean of mean": 1851.063,
        "mean of std": 1198.15311735154,
        "std of mean": 490.45152226392366,
        "std of std": 284.5114400525691
    },
    "turnaround times": {
        "mean of mean": 2354.471,
        "mean of std": 1203.545630623031,
        "std of mean": 570.3442120500566,
        "std of std": 284.78341515652
    },
    "total number of processes": 5000,
    "maximum burst time": "999"
}
```

double wished_range

In [ ]:
```
number_of_tests = 500
wished_range = 2 * 10**3
context_switch_time = 0
tester = SimpleFCFS(n = 1 * 10**2,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                    )
result = tester.runTestNTimes(number_of_tests)
printUniformTestResult(number_of_tests, wished_range, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with Uniform distribution with low 0 and high 2000
{
    "response times": {
        "mean of mean": 48201.1172,
        "mean of std": 28204.025734165996,
        "std of mean": 3276.8594857106336,
        "std of std": 1807.992913635078
    },
    "waiting times": {
        "mean of mean": 48201.1172,
        "mean of std": 28204.025734165996,
        "std of mean": 3276.8594857106336,
        "std of std": 1807.992913635078
    },
    "turnaround times": {
        "mean of mean": 49196.9012,
        "mean of std": 28207.85138247113,
        "std of mean": 3324.8561623322835,
        "std of std": 1806.6778937655843
    },
    "total number of processes": 50000,
```

```
    "maximum burst time": "1999"
}
```

double number of processes

In [ ]:
```
number_of_tests = 500
wished_range = 1 * 10**3
context_switch_time = 0
tester = SimpleFCFS(n = 2 * 10**2,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                    )
result = tester.runTestNTimes(number_of_tests)
printUniformTestResult(number_of_tests, wished_range, result, context_switch_time)
```

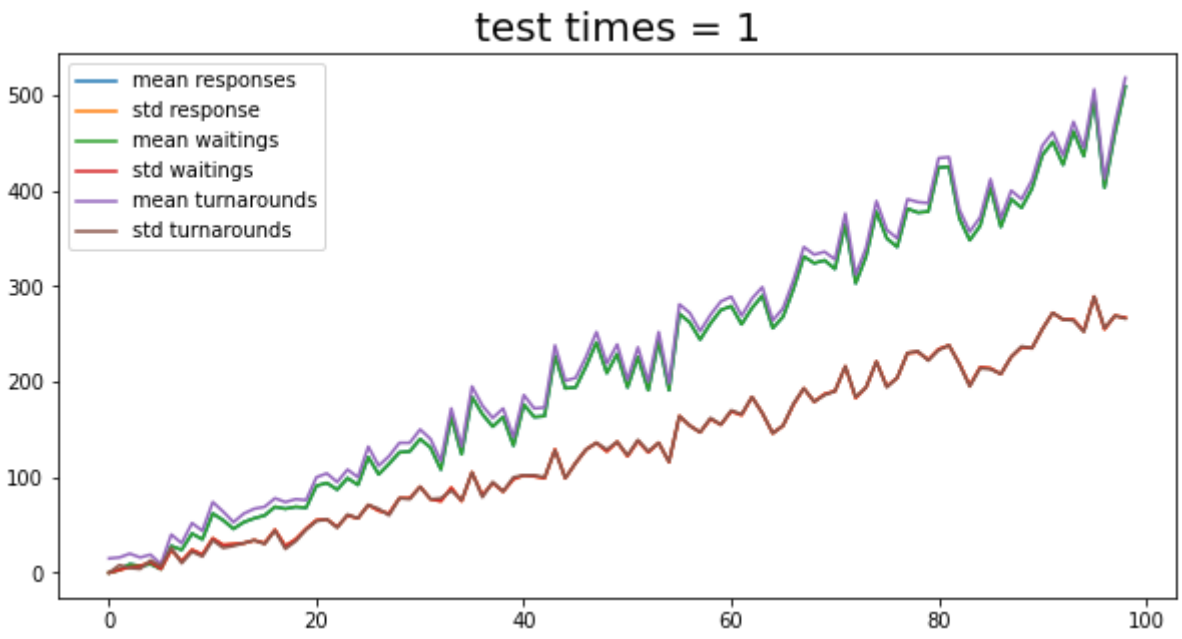
```
ran test 500 times with parameters:
generating numbers with Uniform distribution with low 0 and high 1000
{
    "response times": {
        "mean of mean": 49333.35346,
        "mean of std": 28596.236671737468,
        "std of mean": 2362.4814353513825,
        "std of std": 1332.0668798687007
    },
    "waiting times": {
        "mean of mean": 49333.35346,
        "mean of std": 28596.236671737468,
        "std of mean": 2362.4814353513825,
        "std of std": 1332.0668798687007
    },
    "turnaround times": {
        "mean of mean": 49834.04181,
        "mean of std": 28596.08427976212,
        "std of mean": 2380.4334155694787,
        "std of std": 1331.7412261862428
    },
    "total number of processes": 100000,
    "maximum burst time": "999"
}
```

Everything seems linear! Let's plot

base condition with 1 time test and context switch = 0

In [ ]:
```
wished_range = 10*2
test_times = 1
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleFCFS(n=procces_count,
                        context_switch_time=context_switch_time,
                        randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                        )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('test times = 1',size=20)
```

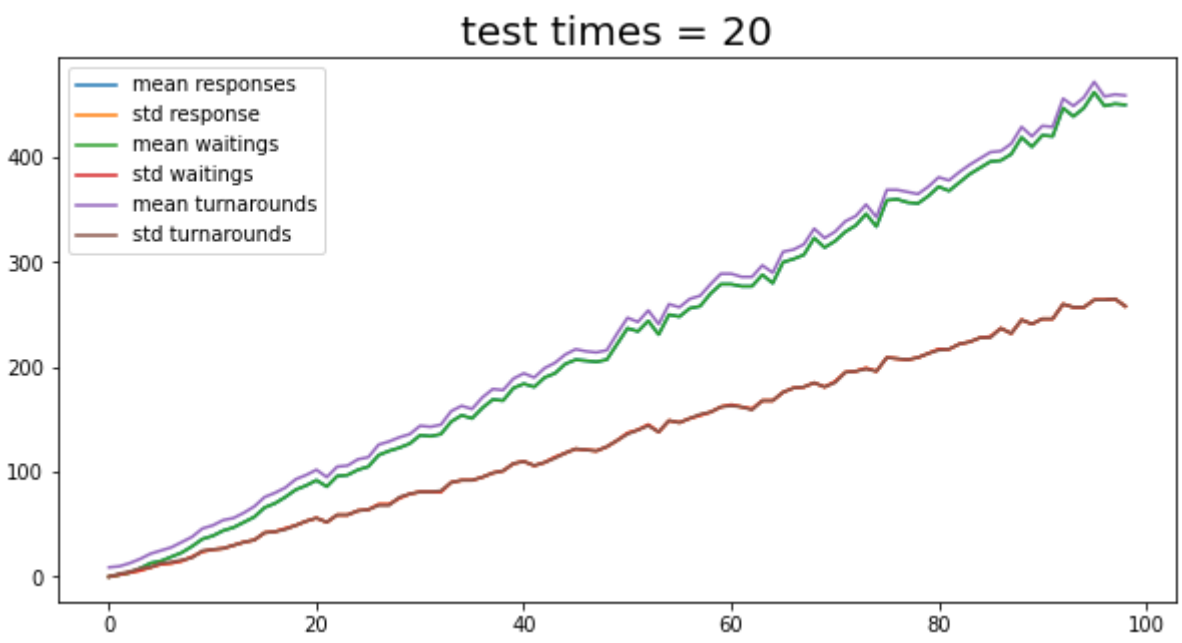
Out[ ]: Text(0.5, 1.0, 'test times = 1')



test times = 20

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleFCFS(n=procces_count,
                        context_switch_time=context_switch_time,
                        randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                    )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('test times = 20',size=20)
```

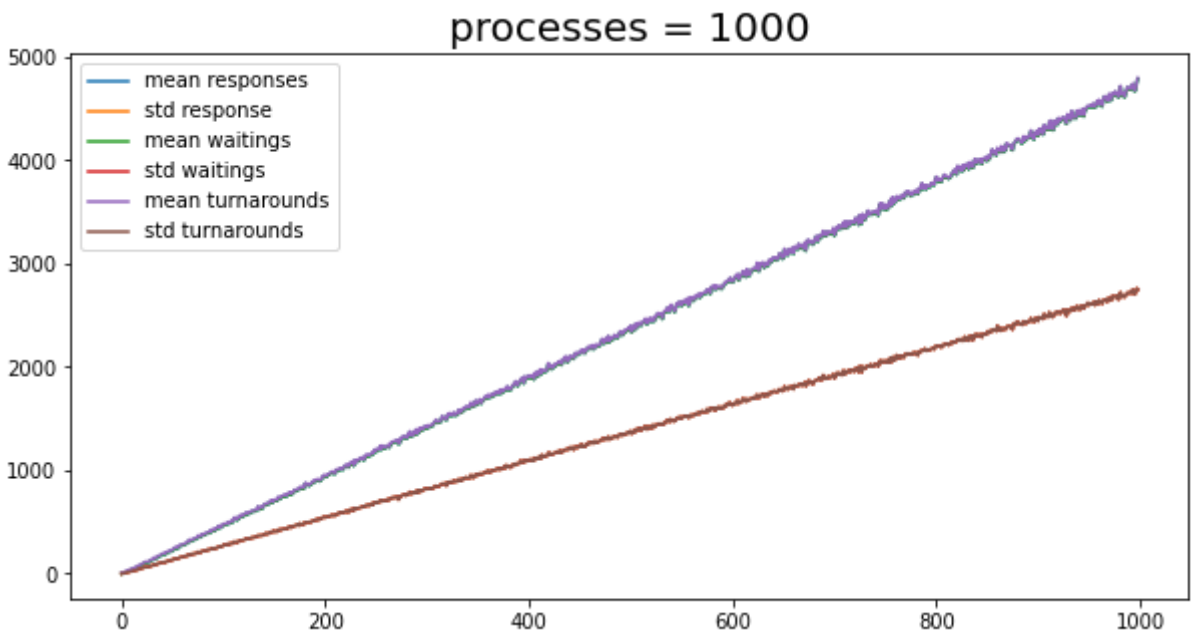
Out[ ]: Text(0.5, 1.0, 'test times = 20')



process count = 1 to 1000

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 1000):
    tester = SimpleFCFS(n=procces_count,
                        context_switch_time=context_switch_time,
                        randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                    )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('processes = 1000',size=20)
```

Out[ ]:
```
Text(0.5, 1.0, 'processes = 1000')
```

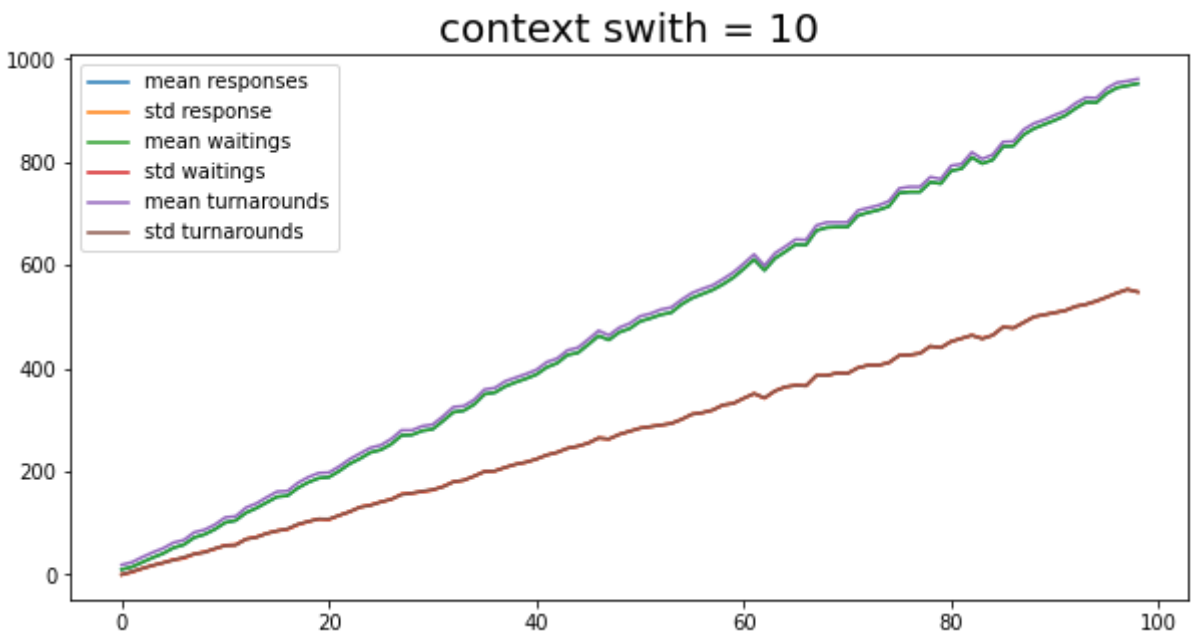


context switch = 10

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 10
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleFCFS(n=procces_count,
                        context_switch_time=context_switch_time,
                        randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                    )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))
```

```
a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('context swith = 10',size=20)
```

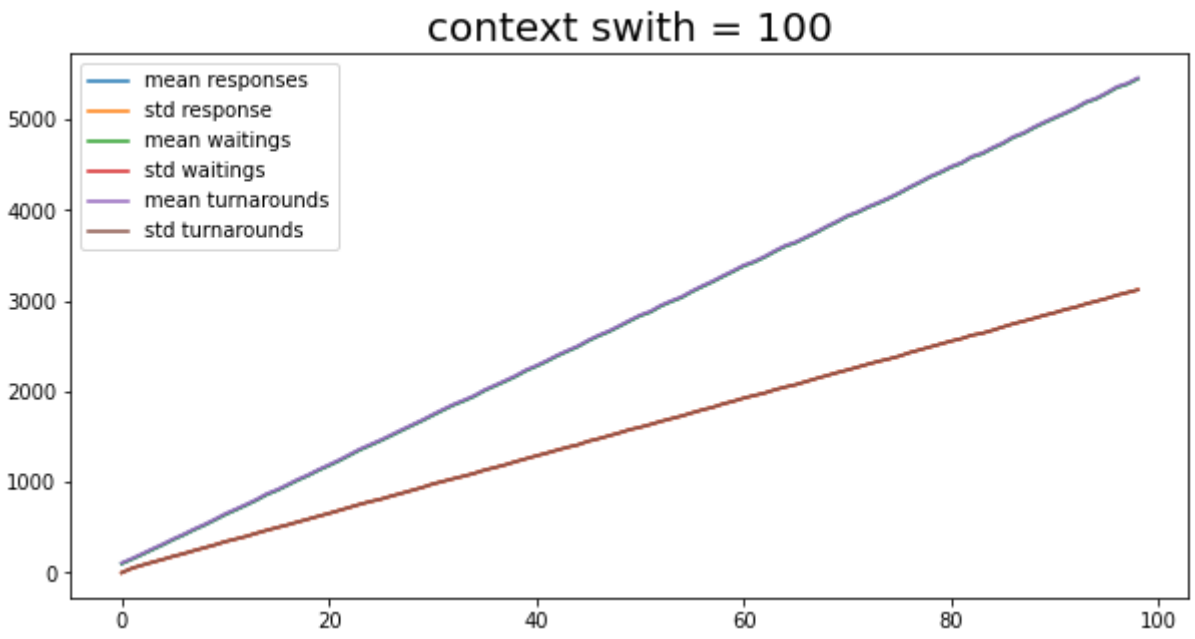
Out[ ]:
```
Text(0.5, 1.0, 'context swith = 10')
```



context switch = 100

In [ ]:
```
wished_range = 10*2
test_times = 20
context_switch_time = 100
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
for procces_count in range(1, 100):
    tester = SimpleFCFS(n=procces_count,
                        context_switch_time=context_switch_time,
                        randomGeneratorFunc=lambda n: getUniformRandomNumber(
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                       )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('context swith = 100',size=20)
```

Out[ ]:
```
Text(0.5, 1.0, 'context swith = 100')
```



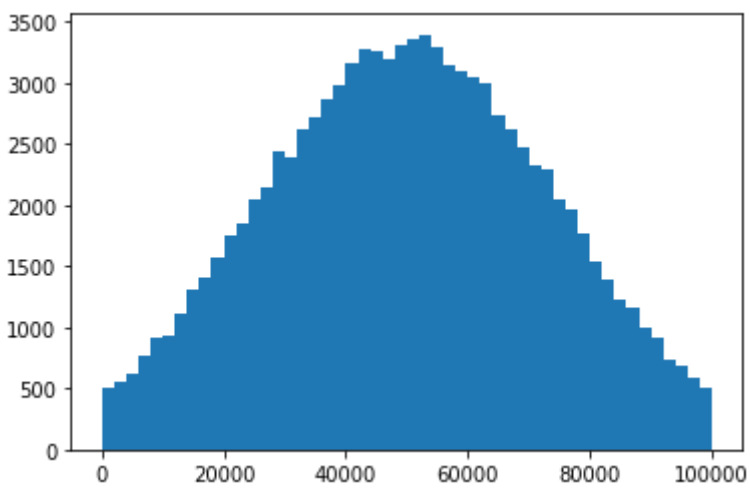
Normal distribution

base condition

In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (4)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleFCFS(n = 10**3,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                        mean=mean,
                                                        std=std,
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 5.0000000000E+04 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 23776904.76961679,
        "mean of std": 13747626.601459544,
        "std of mean": 400842.0577453147,
        "std of std": 217185.8917150406
    },
    "waiting times": {
        "mean of mean": 23776904.76961679,
        "mean of std": 13747626.601459544,
        "std of mean": 400842.0577453147,
        "std of std": 217185.8917150406
    },
    "turnaround times": {
        "mean of mean": 23826911.10454547,
        "mean of std": 13747624.32092067,
        "std of mean": 401427.39648002625,
        "std of std": 217221.09494950745
    },
    "total number of processes": 476960,
    "maximum burst time": "99999"
}
```



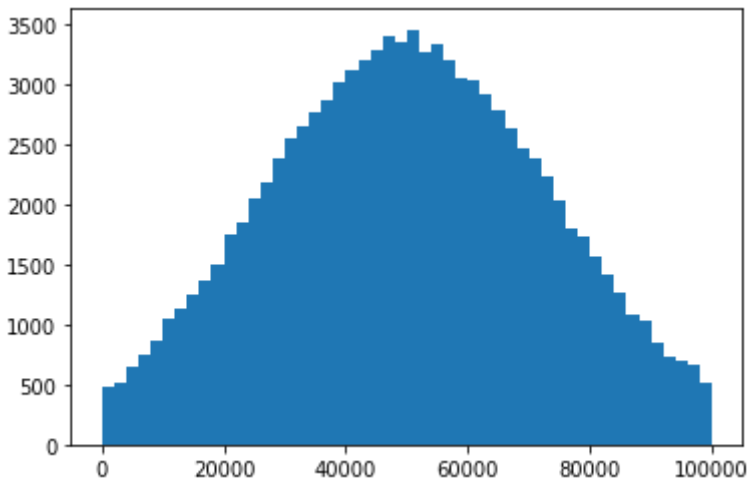
double number of processes

In [ ]:

```
number_of_tests = 500
wished_range = 10**5
```

```
mean = (4)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleFCFS(n = 2* 10**3,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                            mean=mean,
                                                            std=std,
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 5.0000000000E+04 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 47673612.930971235,
        "mean of std": 27530360.78433979,
        "std of mean": 592692.1530757706,
        "std of std": 331260.49509849475
    },
    "waiting times": {
        "mean of mean": 47673612.930971235,
        "mean of std": 27530360.78433979,
        "std of mean": 592692.1530757706,
        "std of std": 331260.49509849475
    },
    "turnaround times": {
        "mean of mean": 47723640.689585626,
        "mean of std": 27530346.21432766,
        "std of mean": 593126.0228587574,
        "std of std": 331272.75679949
    },
    "total number of processes": 954274,
    "maximum burst time": "99999"
}
```

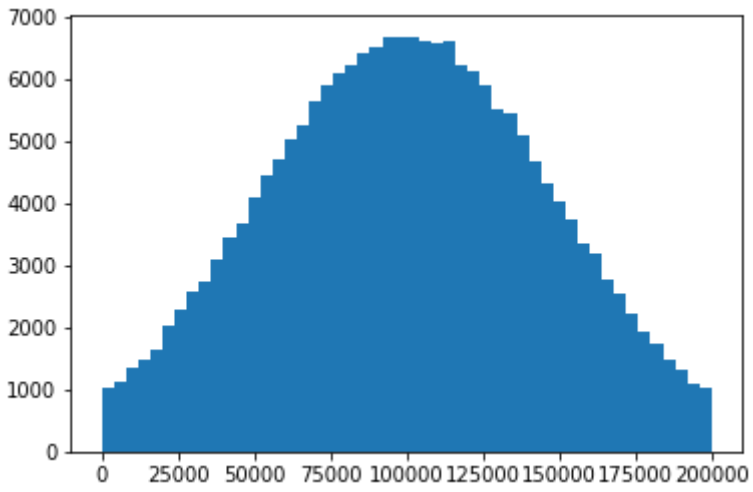


In [ ]:

```
number_of_tests = 500
wished_range = 2 * 10**5
mean = (4)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleFCFS(n = 10**3,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                            mean=mean,
                                                            std=std,
                                                            in_range=wished_range,
                                                            shape=(n, 2)
                                                        )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 1.0000000000E+05 and std of 5.0000000000E+04
{
    "response times": {
        "mean of mean": 47557768.96445417,
        "mean of std": 27522540.070274,
        "std of mean": 800150.202303578,
        "std of std": 464188.7168361254
    },
    "waiting times": {
        "mean of mean": 47557768.96445417,
        "mean of std": 27522540.070274,
        "std of mean": 800150.202303578,
        "std of std": 464188.7168361254
    },
    "turnaround times": {
        "mean of mean": 47657798.093830995,
```

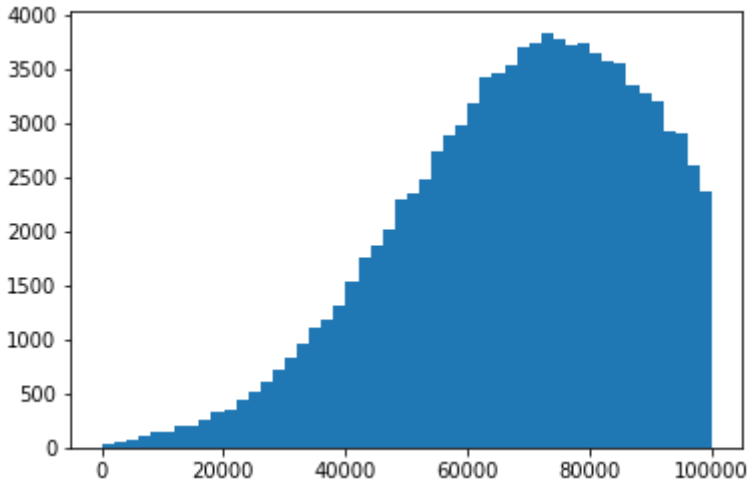
```
        "mean of std": 27522630.410513252,
        "std of mean": 801389.0971602619,
        "std of std": 464309.95660677773
    },
    "total number of processes": 477178,
    "maximum burst time": "199993"
}
```



In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (6)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleFCFS(n = 10**3,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                        mean=mean,
                                                        std=std,
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 7.5000000000E+04 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 28408171.037347123,
        "mean of std": 16444728.72279114,
        "std of mean": 425487.5651351761,
        "std of std": 245132.2703946267
    },
    "waiting times": {
        "mean of mean": 28408171.037347123,
        "mean of std": 16444728.72279114,
        "std of mean": 425487.5651351761,
        "std of std": 245132.2703946267
    },
    "turnaround times": {
        "mean of mean": 28476046.30738125,
        "mean of std": 16444738.630103093,
        "std of mean": 425917.8569598253,
        "std of std": 245156.5336915781
    },
    "total number of processes": 420010,
    "maximum burst time": "99999"
}
```

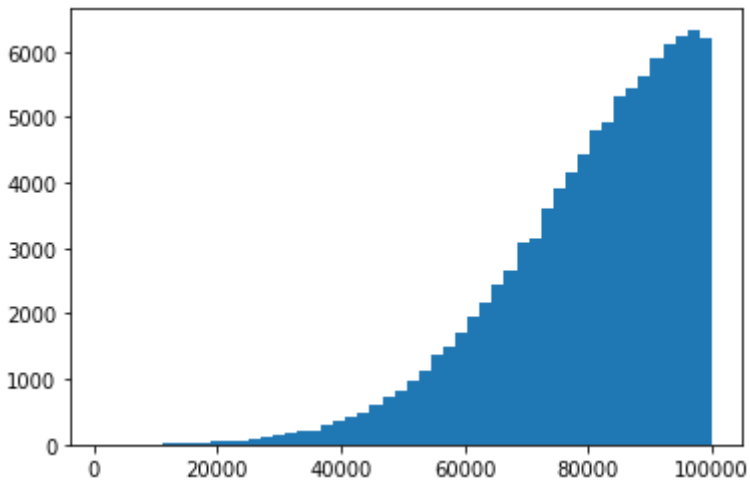


In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (8)*(wished_range)/8
std = 2*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleFCFS(n = 10**3,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getNormalRandomNumber(
```

```
                                                                    mean=mean,
                                                                    std=std,
                                                                    in_range=wished_range,
                                                                    shape=(n, 2)
                                                                )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

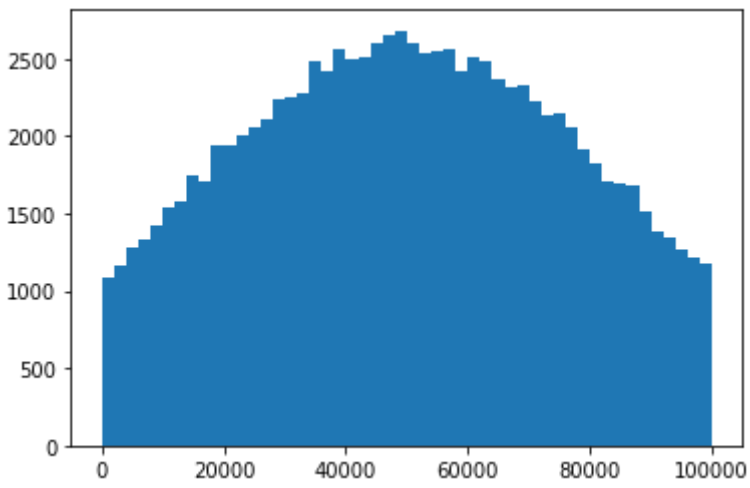
```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 1.0000000000E+05 and std of 2.5000000000E+04
{
    "response times": {
        "mean of mean": 19895345.588927537,
        "mean of std": 11532693.697948268,
        "std of mean": 483066.49244107987,
        "std of std": 272916.2571207285
    },
    "waiting times": {
        "mean of mean": 19895345.588927537,
        "mean of std": 11532693.697948268,
        "std of mean": 483066.49244107987,
        "std of std": 272916.2571207285
    },
    "turnaround times": {
        "mean of mean": 19975434.352681365,
        "mean of std": 11532701.863930976,
        "std of mean": 483285.90291043045,
        "std of std": 272867.6730952919
    },
    "total number of processes": 249675,
    "maximum burst time": "99999"
}
```



In [ ]:

```
number_of_tests = 500
wished_range = 10**5
mean = (4)*(wished_range)/8
std = 3*(wished_range)/8
plt.hist(getNormalRandomNumber(mean=mean, std=std, in_range=wished_range, shape=(20, wished_range))[0], bins=50)
context_switch_time = 0
tester = SimpleFCFS(n = 10**3,
                    context_switch_time=context_switch_time,
                    randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                                    mean=mean,
                                                                    std=std,
                                                                    in_range=wished_range,
                                                                    shape=(n, 2)
                                                                )
                    )
result = tester.runTestNTimes(number_of_tests)
printNormalTestResult(number_of_tests, mean, std, result, context_switch_time)
```

```
ran test 500 times with parameters:
generating numbers with normal distribution with mean 5.0000000000E+04 and std of 3.7500000000E+04
{
    "response times": {
        "mean of mean": 20334540.2398454,
        "mean of std": 11754854.308926372,
        "std of mean": 472127.29900380695,
        "std of std": 259844.14504930505
    },
    "waiting times": {
        "mean of mean": 20334540.2398454,
        "mean of std": 11754854.308926372,
        "std of mean": 472127.29900380695,
        "std of std": 259844.14504930505
    },
    "turnaround times": {
        "mean of mean": 20384464.299862236,
        "mean of std": 11754797.605843373,
        "std of mean": 472831.03855655994,
        "std of std": 259809.9326140212
    },
    "total number of processes": 408536,
    "maximum burst time": "99999"
}
```



In [ ]:

```
wished_range = 10*2
test_times = 20
context_switch_time = 0
mean_responses =[]
std_responses =[]
mean_waitings = []
std_waitings = []
mean_turnarounds = []
std_turnarounds = []
std = wished_range / 4
for mean in range(1, wished_range):
    tester = SimpleFCFS(n=100,
                        context_switch_time=context_switch_time,
                        randomGeneratorFunc=lambda n: getNormalRandomNumber(
                                                        mean=mean,
                                                        std=std,
                                                        in_range=wished_range,
                                                        shape=(n, 2)
                                                    )
                    )

    final_result = tester.runTestNTimes(test_times)
    mean_responses.append(int(final_result['response times']['mean of mean']))
    std_responses.append(int(final_result['response times']['mean of std']))
    mean_waitings.append(int(final_result['waiting times']['mean of mean']))
    std_waitings.append(int(final_result['waiting times']['mean of std']))
    mean_turnarounds.append(int(final_result['turnaround times']['mean of mean']))
    std_turnarounds.append(int(final_result['turnaround times']['mean of std']))

a = pd.DataFrame({
                    'mean responses':mean_responses,
                    'std response': std_responses,

                    'mean waitings':mean_waitings,
                    'std waitings':std_waitings,

                    'mean turnarounds':mean_turnarounds,
                    'std turnarounds':std_turnarounds

                  })
a.plot(figsize=(10,5), fontsize=10)
plt.legend(loc=2, prop={'size': 10})
plt.title('changing mean',size=20)
plt.xlabel('mean')
```

Out[ ]:

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