async()和future<>：

async()：提供一个接口，让一个callable object，若是可能的话在后台运行，成为一个独立线程。

class future<>：允许你等待线程结束并获取其结果。

例：

int DoSomething(char c)

{

*default\_random\_engine* dre(c);

*uniform\_int\_distribution*<int> id(10, 1000);

// loop to print character after a random period of time

for (int i = 0; i < 10; ++i)

{

*this\_thread*::*sleep\_for*(*chrono*::*milliseconds*(id(dre)));

*cout*.*put*(c).*flush*();

}

return c;

}

int Func1()

{

return DoSomething('.');

}

int Func2()

{

return DoSomething('+');

}

int main(int argc, char\* argv[])

{

*cout* << "starting Func1() in background"

<< " and Func2() in foreground:" << *endl*;

// start Func1() asynchronously

// (now or later or never)

// 尝试将Func1异步启动于

// 一个分离线程内

*future*<int> result1(*async*(Func1));

// call Func2() synchronously

int result2 = Func2();

// print result(wait for Func1()

// to finish and add its result to

// result2

int result = result1.*get*() + result2;

*cout* << "\nresult of Func1() + Func2(): " << result

<< *endl*;

return 0;

}

输出为：

starting Func1() in background and Func2() in foreground:

.++...+...+..++.++++

result of Func1() + Func2(): 89

从输出结果.和+交错出现可以看出Func1是异步运行的。