异步客户端和服务端，工人多进程测试

工人进程：

// work\_abc.cpp: 定义控制台应用程序的入口点。

//

#include "stdafx.h"

#include "zmq.h"

#include "zmq\_utils.h"

#include "zhelper.h"

int main()

{

void \*ctx = zmq\_ctx\_new();

void \*worker = zmq\_socket(ctx, ZMQ\_DEALER);

char identity[10] = { 0 };

sprintf(identity, "%04X-%04X", randof(0x10000), randof(0x10000));

zmq\_setsockopt(worker, ZMQ\_IDENTITY, identity, strlen(identity));

// zmq\_connect(worker, "inproc://backend");

zmq\_connect(worker, "ipc:///tmp/feeds/0");

while (1) {

zmq\_msg\_t msg\_recv;

zmq\_msg\_init(&msg\_recv);

int len = zmq\_msg\_recv(&msg\_recv, worker, 0);

printf("len ======== %d \n", len);

char identity[100] = { 0 };

strncpy(identity, (char \*)zmq\_msg\_data(&msg\_recv), len);

printf("ident ========== %s\n", identity);

char result[100];

memset(result, 0, 100);

len = zmq\_msg\_recv(&msg\_recv, worker, 0);

strncpy(result, (char \*)zmq\_msg\_data(&msg\_recv), len);

printf("empty ========== %s\n\n", result);

memset(result, 0, 100);

len = zmq\_msg\_recv(&msg\_recv, worker, 0);

strncpy(result, (char \*)zmq\_msg\_data(&msg\_recv), len);

printf("content ========== %s\n\n", result);

zmq\_close(&msg\_recv);

int reply, replies = 2;// rand();

for (reply = 0; reply < replies; reply++) {

Sleep(randof(1000) + 1);

zmq\_send(worker, identity, strlen(identity), ZMQ\_SNDMORE);

zmq\_send(worker, "B nice to meet you", 17, 0);

}

// zmq\_msg\_recv(&msg\_recv, worker, 0);

}

return 0;

}

代理进程：

// worker\_deal.cpp: 定义控制台应用程序的入口点。

//

#include "stdafx.h"

#include "zmq.h"

#include "zmq\_utils.h"

#include "zhelper.h"

#include "json\json.h"

#include <json\config.h>

static DWORD WINAPI server\_worker(void \* context)

{

void \*worker = zmq\_socket(context, ZMQ\_DEALER);

char identity[10] = { 0 };

sprintf(identity, "%04X-%04X", randof(0x10000), randof(0x10000));

zmq\_setsockopt(worker, ZMQ\_IDENTITY, identity, strlen(identity));

zmq\_connect(worker, "inproc://backend");

while (1) {

zmq\_msg\_t msg\_recv;

zmq\_msg\_init(&msg\_recv);

int len = zmq\_msg\_recv(&msg\_recv, worker, 0);

printf("len ======== %d \n", len);

char identity[100] = { 0 };

strncpy(identity, (char \*)zmq\_msg\_data(&msg\_recv),len);

printf("ident ========== %s\n", identity);

char result[100];

memset(result, 0, 100);

len = zmq\_msg\_recv(&msg\_recv, worker, 0);

strncpy(result, (char \*)zmq\_msg\_data(&msg\_recv), len);

printf("empty ========== %s\n\n", result);

memset(result, 0, 100);

len = zmq\_msg\_recv(&msg\_recv, worker, 0);

strncpy(result, (char \*)zmq\_msg\_data(&msg\_recv),len);

printf("content ========== %s\n\n", result);

zmq\_close(&msg\_recv);

int reply, replies = 2;// rand();

for (reply = 0; reply < replies; reply++) {

Sleep(randof(1000) + 1);

zmq\_send(worker,identity , strlen(identity), ZMQ\_SNDMORE);

zmq\_send(worker, "B nice to meet you", 17, 0);

}

// zmq\_msg\_recv(&msg\_recv, worker, 0);

}

}

int main()

{

void \*ctx = zmq\_ctx\_new();

void \*frontend = zmq\_socket(ctx, ZMQ\_ROUTER);

zmq\_bind(frontend, "tcp://192.168.1.137:6001");

void \*backend = zmq\_socket(ctx, ZMQ\_DEALER);

// zmq\_bind(backend, "inproc://backend");

zmq\_bind(backend, "ipc:///tmp/feeds/0");

//int thread\_nbr;

//for (thread\_nbr = 0; thread\_nbr < 1; thread\_nbr++) {

// HANDLE worker;

// DWORD lpThreadId2 = 0;

// worker = CreateThread(NULL, 0, server\_worker, ctx, 0, &lpThreadId2);

//}

#if 1

//初始化轮询集

zmq\_pollitem\_t items[] = {

{ frontend,0,ZMQ\_POLLIN,0 },

{ backend,0,ZMQ\_POLLIN,0 }

};

while (1) {

zmq\_msg\_t message;

int more;

zmq\_poll(items, 2, -1);

if (items[0].revents & ZMQ\_POLLIN) {

zmq\_msg\_init(&message);

zmq\_msg\_recv(&message, frontend, 0);

size\_t more\_size = sizeof(more);

zmq\_getsockopt(frontend, ZMQ\_RCVMORE, &more, &more\_size);

printf("frontend more ================= %d\n", more);

zmq\_msg\_send(&message, backend, more ? ZMQ\_SNDMORE : 0);

zmq\_msg\_close(&message);

if (!more)

break;

}

if (items[1].revents & ZMQ\_POLLIN) {

while (1) {

zmq\_msg\_init(&message);

zmq\_msg\_recv(&message, backend, 0);

size\_t more\_size = sizeof(more);

zmq\_getsockopt(backend, ZMQ\_RCVMORE, &more, &more\_size);

printf("backend more ================= %d\n", more);

zmq\_msg\_send(&message, frontend, more ? ZMQ\_SNDMORE : 0);

zmq\_msg\_close(&message);

if (!more)

break;

}

}

}

#endif

//zmq\_proxy(frontend, backend, NULL);

//while (1);

zmq\_close(frontend);

zmq\_close(backend);

return 0;

}

客户端进程：

// deal\_mode\_cli.cpp: 定义控制台应用程序的入口点。

//

#include "stdafx.h"

#include "zmq.h"

#include "zmq\_utils.h"

#include "zhelper.h"

#include "json\json.h"

#include <json\config.h>

int main()

{

void \*context = zmq\_ctx\_new();

// void \*client = zmq\_socket(context, ZMQ\_REQ);

void \*client = zmq\_socket(context, ZMQ\_DEALER);

char identity[10] = { 0 };

sprintf(identity, "%04X-%04X", randof(0x10000), randof(0x10000));

zmq\_setsockopt(client, ZMQ\_IDENTITY, identity, strlen(identity));

zmq\_connect(client, "tcp://192.168.1.137:6001");

zmq\_pollitem\_t items[] = { {client,0,ZMQ\_POLLIN,0} };

int request\_nbr = 0;

zmq\_msg\_t msg\_send;

while (1) {

int centitick;

for (centitick = 0; centitick < 100; centitick++) {

zmq\_poll(items, 1, 1);

if (items[0].revents & ZMQ\_POLLIN) {

zmq\_msg\_t msg\_recv;

zmq\_msg\_init(&msg\_recv);

zmq\_msg\_recv(&msg\_recv, client, 0);

printf("recv msg ==== %s\n", (char \*)zmq\_msg\_data(&msg\_recv));

zmq\_msg\_close(&msg\_recv);

}

}

zmq\_msg\_init(&msg\_send);

zmq\_send(client, "", 0, ZMQ\_SNDMORE);

char str[20] = { 0 };

sprintf(str, "request #%d", ++request\_nbr);

memcpy(zmq\_msg\_data(&msg\_send), str, strlen(str));

// zmq\_msg\_send(&msg\_send, client, 0);

zmq\_send(client, str, strlen(str), 0);

Sleep(5000);

zmq\_msg\_close(&msg\_send);

}

zmq\_ctx\_destroy(context);

return 0;

}

代理frontend 可以收到消息，但是work\_abc收不到消息，阻塞等待中，不知是进程通信不行还是这种代理不支持跨进程，需要验证